

# Keysight InfiniiVision 3000T X-Series Oscilloscopes



Programmer's  
Guide

# Notices

© Keysight Technologies, Inc. 2005-2016

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, Inc. as governed by United States and international copyright laws.

## Revision

Version 04.07.0000

## Edition

April 20, 2016

Available in electronic format only

Published by:

Keysight Technologies, Inc.  
1900 Garden of the Gods Road  
Colorado Springs, CO 80907 USA

## Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Keysight disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Keysight shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Keysight and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

## Technology License

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

## U.S. Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at [www.keysight.com/find/sweula](http://www.keysight.com/find/sweula). The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of these rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

## Safety Notices

### CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

### WARNING

A **WARNING** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a **WARNING** notice until the indicated conditions are fully understood and met.

# In This Book

This book is your guide to programming the 3000T X-Series oscilloscopes:

**Table 1** InfiniiVision 3000T X-Series Oscilloscope Models, Bandwidths, Sample Rates

Bandwidth	100 MHz	200 MHz	350 MHz	500 MHz	1 GHz
Sample Rate (interleaved, non-interleaved)	5 GSa/s, 2.5 GSa/s				
2-Channel + 16 Logic Channels MSO	MSO-X 3012T	MSO-X 3022T	MSO-X 3032T	MSO-X 3052T	MSO-X 3102T
4-Channel + 16 Logic Channels MSO	MSO-X 3014T	MSO-X 3024T	MSO-X 3034T	MSO-X 3054T	MSO-X 3104T
2-Channel DSO	DSO-X 3012T	DSO-X 3022T	DSO-X 3032T	DSO-X 3052T	DSO-X 3102T
4-Channel DSO	DSO-X 3014T	DSO-X 3024T	DSO-X 3034T	DSO-X 3054T	DSO-X 3104T

The first few chapters describe how to set up and get started:

- **Chapter 1**, “What’s New,” starting on page 35, describes programming command changes in the latest version of oscilloscope software.
- **Chapter 2**, “Setting Up,” starting on page 47, describes the steps you must take before you can program the oscilloscope.
- **Chapter 3**, “Getting Started,” starting on page 55, gives a general overview of oscilloscope program structure and shows how to program the oscilloscope using a few simple examples.
- **Chapter 4**, “Commands Quick Reference,” starting on page 69, is a brief listing of the 3000T X-Series oscilloscope commands and syntax.

The next chapters provide reference information on common commands, root level commands, other subsystem commands, and error messages:

- **Chapter 5**, “Common (\*) Commands,” starting on page 177, describes commands defined by the IEEE 488.2 standard that are common to all instruments.
- **Chapter 6**, “Root (:) Commands,” starting on page 203, describes commands that reside at the root level of the command tree and control many of the basic functions of the oscilloscope.
- **Chapter 7**, “:ACQuire Commands,” starting on page 243, describes commands for setting the parameters used when acquiring and storing data.
- **Chapter 8**, “:BUS<n> Commands,” starting on page 257, describes commands that control all oscilloscope functions associated with the digital channels bus display.
- **Chapter 9**, “:CALibrate Commands,” starting on page 267, describes utility commands for determining the state of the calibration factor protection button.

- **Chapter 10**, “:`:CHANnel<n>` Commands,” starting on page 279, describes commands that control all oscilloscope functions associated with individual analog channels or groups of channels.
- **Chapter 11**, “:`:COUNter` Commands,” starting on page 301, describes commands that control the optional DSOXDVMCTR counter analysis feature.
- **Chapter 12**, “:`:DEMO` Commands,” starting on page 313, describes commands that control the education kit (Option EDU) demonstration signals that can be output on the oscilloscope’s Demo 1 and Demo 2 terminals.
- **Chapter 13**, “:`:DIGital<d>` Commands,” starting on page 321, describes commands that control all oscilloscope functions associated with individual digital channels.
- **Chapter 14**, “:`:DISPlay` Commands,” starting on page 329, describes commands that control how waveforms, graticule, and text are displayed and written on the screen.
- **Chapter 15**, “:`:DVM` Commands,” starting on page 347, describes commands that control the optional DSOXDVM digital voltmeter analysis feature.
- **Chapter 16**, “:`:EXTernal Trigger` Commands,” starting on page 353, describes commands that control the input characteristics of the external trigger input.
- **Chapter 17**, “:`:FFT` Commands,” starting on page 359, describes commands that control the FFT function in the oscilloscope.
- **Chapter 18**, “:`:FUNCTION<m>` Commands,” starting on page 379, describes commands that control math waveforms.
- **Chapter 19**, “:`:HARDcopy` Commands,” starting on page 417, describes commands that set and query the selection of hardcopy device and formatting options.
- **Chapter 20**, “:`:LISTer` Commands,” starting on page 435, describes commands that turn on/off the Lister display for decoded serial data and get the Lister data.
- **Chapter 21**, “:`:MARKer` Commands,” starting on page 439, describes commands that set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors).
- **Chapter 22**, “:`:MEASure` Commands,” starting on page 461, describes commands that select automatic measurements (and control markers).
- **Chapter 23**, “:`:MEASure Power` Commands,” starting on page 543, describes measurement commands that are available when the DSOX4PWR power measurements and analysis application is licensed and enabled.
- **Chapter 24**, “:`:MTEST` Commands,” starting on page 567, describes commands that control the mask test features provided with Option LMT.
- **Chapter 25**, “:`:POD` Commands,” starting on page 601, describes commands that control all oscilloscope functions associated with groups of digital channels.
- **Chapter 26**, “:`:POWER` Commands,” starting on page 607, describes commands that control the DSOX4PWR power measurement application.

- **Chapter 27**, “:RECall Commands,” starting on page 683, describes commands that recall previously saved oscilloscope setups, reference waveforms, or masks.
- **Chapter 28**, “:SAVE Commands,” starting on page 693, describes commands that save oscilloscope setups, screen images, and data.
- **Chapter 29**, “:SBUS<n> Commands,” starting on page 723, describes commands that control oscilloscope functions associated with the serial decode bus and serial triggering.
- **Chapter 30**, “:SEARch Commands,” starting on page 919, describes commands that control oscilloscope functions associated with searching for waveform events.
- **Chapter 31**, “:SYSTem Commands,” starting on page 1011, describes commands that control basic system functions of the oscilloscope.
- **Chapter 32**, “:TIMEbase Commands,” starting on page 1035, describes commands that control all horizontal sweep functions.
- **Chapter 33**, “:TRIGger Commands,” starting on page 1047, describes commands that control the trigger modes and parameters for each trigger type.
- **Chapter 34**, “:WAVeform Commands,” starting on page 1145, describes commands that provide access to waveform data.
- **Chapter 35**, “:WGEN<w> Commands,” starting on page 1181, describes commands that control waveform generator (Option WGN) functions and parameters.
- **Chapter 36**, “:WMEMory<r> Commands,” starting on page 1223, describes commands that control reference waveforms.
- **Chapter 37**, “Obsolete and Discontinued Commands,” starting on page 1233, describes obsolete commands which still work but have been replaced by newer commands and discontinued commands which are no longer supported.
- **Chapter 38**, “Error Messages,” starting on page 1291, lists the instrument error messages that can occur while programming the oscilloscope.

The command descriptions in this reference show upper and lowercase characters. For example, :AUToscale indicates that the entire command name is :AUTOSCALE. The short form, :AUT, is also accepted by the oscilloscope.

Then, there are chapters that describe programming topics and conceptual information in more detail:

- **Chapter 39**, “Status Reporting,” starting on page 1299, describes the oscilloscope’s status registers and how to check the status of the instrument.
- **Chapter 40**, “Synchronizing Acquisitions,” starting on page 1323, describes how to wait for acquisitions to complete before querying measurement results or performing other operations with the captured data.
- **Chapter 41**, “More About Oscilloscope Commands,” starting on page 1333, contains additional information about oscilloscope programming commands.

Finally, there is a chapter that contains programming examples:

- **Chapter 42**, "Programming Examples," starting on page 1343.

**Mixed-Signal Oscilloscope Channel Differences** Because both the "analog channels only" oscilloscopes (DSO models) and the mixed-signal oscilloscopes (MSO models) have analog channels, topics that describe analog channels refer to all oscilloscope models. Whenever a topic describes digital channels, that information applies only to the mixed-signal oscilloscope models.

**See Also**

- For more information on using the SICL, VISA, and VISA COM libraries in general, see the documentation that comes with the Keysight IO Libraries Suite.
- For information on controller PC interface configuration, see the documentation for the interface card used (for example, the Keysight 82350B GPIB interface).
- For information on oscilloscope front-panel operation, see the *User's Guide*.
- For detailed connectivity information, refer to the *Keysight Technologies USB/LAN/GPIB Connectivity Guide*. For a printable electronic copy of the *Connectivity Guide*, direct your Web browser to [www.keysight.com](http://www.keysight.com) and search for "Connectivity Guide".
- For the latest versions of this and other manuals, see:  
<http://www.keysight.com/find/3000T-X-Series-manual>

# Contents

## In This Book / 3

### 1 What's New

What's New in Version 4.07 / 36
What's New in Version 4.06 / 38
What's New in Version 4.05 / 39
Version 4.00 at Introduction / 41
Command Differences From 4000 X-Series Oscilloscopes / 42

### 2 Setting Up

Step 1. Install Keysight IO Libraries Suite software / 48
Step 2. Connect and set up the oscilloscope / 49
Using the USB (Device) Interface / 49
Using the LAN Interface / 49
Using the GPIB Interface / 50
Step 3. Verify the oscilloscope connection / 51

### 3 Getting Started

Basic Oscilloscope Program Structure / 56
Initializing / 56
Capturing Data / 56
Analyzing Captured Data / 57

Programming the Oscilloscope / 58
Referencing the IO Library / 58
Opening the Oscilloscope Connection via the IO Library / 59
Initializing the Interface and the Oscilloscope / 59
Using :AUToscale to Automate Oscilloscope Setup / 60
Using Other Oscilloscope Setup Commands / 60
Capturing Data with the :DIGitize Command / 61
Reading Query Responses from the Oscilloscope / 63
Reading Query Results into String Variables / 64
Reading Query Results into Numeric Variables / 64
Reading Definite-Length Block Query Response Data / 64
Sending Multiple Queries and Reading Results / 65
Checking Instrument Status / 66
Other Ways of Sending Commands / 67
Telnet Sockets / 67
Sending SCPI Commands Using Browser Web Control / 67

## 4 Commands Quick Reference

Command Summary / 70
Syntax Elements / 174
Number Format / 174
<NL> (Line Terminator) / 174
[ ] (Optional Syntax Terms) / 174
{ } (Braces) / 174
::= (Defined As) / 174
< > (Angle Brackets) / 175
... (Ellipsis) / 175
n,...,p (Value Ranges) / 175
d (Digits) / 175
Quoted ASCII String / 175
Definite-Length Block Response Data / 175

## 5 Common (\*) Commands

*CLS (Clear Status) / 181
*ESE (Standard Event Status Enable) / 182
*ESR (Standard Event Status Register) / 184
*IDN (Identification Number) / 186
*LRN (Learn Device Setup) / 187
*OPC (Operation Complete) / 188
*OPT (Option Identification) / 189

\*RCL (Recall) / 191  
\*RST (Reset) / 192  
\*SAV (Save) / 195  
\*SRE (Service Request Enable) / 196  
\*STB (Read Status Byte) / 198  
\*TRG (Trigger) / 200  
\*TST (Self Test) / 201  
\*WAI (Wait To Continue) / 202

## 6 Root (:) Commands

:ACTivity / 207  
:AER (Arm Event Register) / 208  
:AUToscale / 209  
:AUToscale:AMODE / 211  
:AUToscale:CHANnels / 212  
:AUToscale:FDEBug / 213  
:BLANK / 214  
:DIGItize / 215  
:HWEnable (Hardware Event Enable Register) / 217  
:HWERegister:CONDition (Hardware Event Condition Register) / 219  
:HWERegister[:EVENT] (Hardware Event Event Register) / 220  
:MTEnable (Mask Test Event Enable Register) / 221  
:MTERegister[:EVENT] (Mask Test Event Event Register) / 223  
:OPEE (Operation Status Enable Register) / 225  
:OPERegister:CONDition (Operation Status Condition Register) / 227  
:OPERegister[:EVENT] (Operation Status Event Register) / 229  
:OVLenable (Overload Event Enable Register) / 231  
:OVLRegister (Overload Event Register) / 233  
:PRInt / 235  
:RUN / 236  
:SERial / 237  
:SINGle / 238  
:STATus / 239  
:STOP / 240  
:TER (Trigger Event Register) / 241  
:VIEW / 242

## 7 :ACQuire Commands

:ACQuire:COMplete / 245  
:ACQuire:COUNt / 246  
:ACQuire:MODE / 247

:ACQuire:POINts / 248  
:ACQuire:SEGmented:ANALyze / 249  
:ACQuire:SEGmented:COUNt / 250  
:ACQuire:SEGmented:INDex / 251  
:ACQuire:SRATe / 254  
:ACQuire:TYPE / 255

## 8 :BUS<n> Commands

:BUS<n>:BIT<m> / 259  
:BUS<n>:BITS / 260  
:BUS<n>:CLEar / 262  
:BUS<n>:DISPlay / 263  
:BUS<n>:LABel / 264  
:BUS<n>:MASK / 265

## 9 :CALibrate Commands

:CALibrate:DATE / 269  
:CALibrate:LABel / 270  
:CALibrate:OUTPut / 271  
:CALibrate:PROTected / 273  
:CALibrate:STARt / 274  
:CALibrate:STATus / 275  
:CALibrate:TEMPerature / 276  
:CALibrate:TIME / 277

## 10 :CHANnel<n> Commands

:CHANnel<n>:BWLimit / 282  
:CHANnel<n>:COUPLing / 283  
:CHANnel<n>:DISPlay / 284  
:CHANnel<n>:IMPedance / 285  
:CHANnel<n>:INVert / 286  
:CHANnel<n>:LABel / 287  
:CHANnel<n>:OFFSet / 288  
:CHANnel<n>:PROBe / 289  
:CHANnel<n>:PROBe:HEAD[:TYPE] / 290  
:CHANnel<n>:PROBe:ID / 291  
:CHANnel<n>:PROBe:MMODel / 292  
:CHANnel<n>:PROBe:SKEW / 293  
:CHANnel<n>:PROBe:STYPe / 294  
:CHANnel<n>:PROTection / 295  
:CHANnel<n>:RANGe / 296

:CHANnel<n>:SCALe / 297  
:CHANnel<n>:UNITS / 298  
:CHANnel<n>:VERNier / 299

## 11 :COUNter Commands

:COUNter:CURRent / 303  
:COUNter:ENABLE / 304  
:COUNter:MODE / 305  
:COUNter:NDIGits / 306  
:COUNter:SOURce / 307  
:COUNter:TOTalize:CLEar / 308  
:COUNter:TOTalize:GATE:ENABLE / 309  
:COUNter:TOTalize:GATE:POLarity / 310  
:COUNter:TOTalize:GATE:SOURce / 311  
:COUNter:TOTalize:SLOPe / 312

## 12 :DEMO Commands

:DEMO:FUNCtion / 314  
:DEMO:FUNCtion:PHASe:PHASe / 318  
:DEMO:OUTPut / 319

## 13 :DIGItal<d> Commands

:DIGItal<d>:DISPlay / 323  
:DIGItal<d>:LABel / 324  
:DIGItal<d>:POsition / 325  
:DIGItal<d>:SIZE / 326  
:DIGItal<d>:THRehold / 327

## 14 :DISPlay Commands

:DISPlay:ANNotation<n> / 332  
:DISPlay:ANNotation<n>:BACKground / 333  
:DISPlay:ANNotation<n>:COLor / 334  
:DISPlay:ANNotation<n>:TEXT / 335  
:DISPlay:ANNotation<n>:X1Position / 336  
:DISPlay:ANNotation<n>:Y1Position / 337  
:DISPlay:CLEar / 338  
:DISPlay:DATA / 339  
:DISPlay:INTensity:WAVeform / 340  
:DISPlay:LABel / 341  
:DISPlay:LABList / 342  
:DISPlay:MENU / 343

:DISPlay:SIDebar / 344  
:DISPlay:PERsistence / 345  
:DISPlay:VECTors / 346

## 15 :DVM Commands

:DVM:ARAnge / 348  
:DVM:CURREnt / 349  
:DVM:ENABLE / 350  
:DVM:MODE / 351  
:DVM:SOURce / 352

## 16 :EXTernal Trigger Commands

:EXTernal:BWLImit / 354  
:EXTernal:PROBe / 355  
:EXTernal:RANGE / 356  
:EXTernal:UNITS / 357

## 17 :FFT Commands

:FFT:AVERage:COUNT / 361  
:FFT:CENTer / 362  
:FFT:CLEar / 363  
:FFT:DISPlay / 364  
:FFT:DMODe / 365  
:FFT:FREQuency:STARt / 367  
:FFT:FREQuency:STOP / 368  
:FFT:GATE / 369  
:FFT:OFFSet / 370  
:FFT:RANGE / 371  
:FFT:REFERENCE / 372  
:FFT:SCALe / 373  
:FFT:SOURce1 / 374  
:FFT:SPAN / 375  
:FFT:VTPe / 376  
:FFT:WINDOW / 377

## 18 :FUNCTION<m> Commands

:FUNCTION<m>:AVERage:COUNT / 384  
:FUNCTION<m>:BUS:CLOCK / 385  
:FUNCTION<m>:BUS:SLOPe / 386  
:FUNCTION<m>:BUS:YINCrement / 387  
:FUNCTION<m>:BUS:YORigin / 388

:FUNCTION<m>:BUS:YUNits / 389  
:FUNCTION<m>:CLEar / 390  
:FUNCTION<m>:DISPlay / 391  
:FUNCTION<m>[:FFT]:CENTer / 392  
:FUNCTION<m>[:FFT]:FREQuency:STARt / 393  
:FUNCTION<m>[:FFT]:FREQuency:STOP / 394  
:FUNCTION<m>[:FFT]:GATE / 395  
:FUNCTION<m>[:FFT]:SPAN / 396  
:FUNCTION<m>[:FFT]:VTYPe / 397  
:FUNCTION<m>[:FFT]:WINDOW / 398  
:FUNCTION<m>:FREQuency:HIGHpass / 399  
:FUNCTION<m>:FREQuency:LOWPass / 400  
:FUNCTION<m>:INTegrate:IOFFset / 401  
:FUNCTION<m>:LINear:GAIN / 402  
:FUNCTION<m>:LINear:OFFSet / 403  
:FUNCTION<m>:OFFSet / 404  
:FUNCTION<m>:OPERation / 405  
:FUNCTION<m>:RANGe / 409  
:FUNCTION<m>:REFERence / 410  
:FUNCTION<m>:SCALE / 411  
:FUNCTION<m>:SMOoth:POINTs / 412  
:FUNCTION<m>:SOURce1 / 413  
:FUNCTION<m>:SOURce2 / 415  
:FUNCTION<m>:TRENd:MEASurement / 416

## 19 :HARDcopy Commands

:HARDcopy:AREA / 419  
:HARDcopy:APRinter / 420  
:HARDcopy:FACTors / 421  
:HARDcopy:FFEed / 422  
:HARDcopy:INKSaver / 423  
:HARDcopy:LAYout / 424  
:HARDcopy:NETWork:ADDRess / 425  
:HARDcopy:NETWork:APPLy / 426  
:HARDcopy:NETWork:DOMain / 427  
:HARDcopy:NETWork:PASSword / 428  
:HARDcopy:NETWork:SLOT / 429  
:HARDcopy:NETWork:USERname / 430  
:HARDcopy:PAlette / 431  
:HARDcopy:PRINter:LIST / 432  
:HARDcopy:STARt / 433

## 20 :LISTer Commands

:LISTer:DATA / 436  
:LISTer:DISPlay / 437  
:LISTer:REFerence / 438

## 21 :MARKer Commands

:MARKer:DYDX / 442  
:MARKer:MODE / 443  
:MARKer:X1:DISPlay / 444  
:MARKer:X1Position / 445  
:MARKer:X1Y1source / 446  
:MARKer:X2:DISPlay / 447  
:MARKer:X2Position / 448  
:MARKer:X2Y2source / 449  
:MARKer:XDELta / 450  
:MARKer:XUNits / 451  
:MARKer:XUNits:USE / 452  
:MARKer:Y1:DISPlay / 453  
:MARKer:Y1Position / 454  
:MARKer:Y2:DISPlay / 455  
:MARKer:Y2Position / 456  
:MARKer:YDELta / 457  
:MARKer:YUNits / 458  
:MARKer:YUNits:USE / 459

## 22 :MEASure Commands

:MEASure:ALL / 478  
:MEASure:AREa / 479  
:MEASure:BRATe / 480  
:MEASure:BWIDth / 481  
:MEASure:CLEar / 482  
:MEASure:COUNter / 483  
:MEASure:DEFine / 485  
:MEASure:DELay / 488  
:MEASure:DUAL:CHARge / 490  
:MEASure:DUAL:VAMPLitude / 491  
:MEASure:DUAL:VAVerage / 492  
:MEASure:DUAL:VBASe / 493  
:MEASure:DUAL:VPP / 494  
:MEASure:DUAL:VRMS / 495  
:MEASure:DUTYcycle / 496

:MEASure:FALLtime / 497  
:MEASure:FREQuency / 498  
:MEASure:NDUTy / 499  
:MEASure:NEDGes / 500  
:MEASure:NPULses / 501  
:MEASure:NWIDth / 502  
:MEASure:OVERshoot / 503  
:MEASure:PEDGes / 505  
:MEASure:PERiod / 506  
:MEASure:PHASe / 507  
:MEASure:PPULses / 508  
:MEASure:PRESHoot / 509  
:MEASure:PWIDth / 510  
:MEASure:RESults / 511  
:MEASure:RISetime / 514  
:MEASure:SDEViation / 515  
:MEASure:SHOW / 516  
:MEASure:SOURce / 517  
:MEASure:STATistics / 519  
:MEASure:STATistics:DISPlay / 520  
:MEASure:STATistics:INCRement / 521  
:MEASure:STATistics:MCOut / 522  
:MEASure:STATistics:RESet / 523  
:MEASure:STATistics:RSDeviation / 524  
:MEASure:TEDGe / 525  
:MEASure:TVALue / 527  
:MEASure:VAMPLitude / 529  
:MEASure:VAVerage / 530  
:MEASure:VBASe / 531  
:MEASure:VMAX / 532  
:MEASure:VMIN / 533  
:MEASure:VPP / 534  
:MEASure:VRATio / 535  
:MEASure:VRMS / 536  
:MEASure:VTIMe / 537  
:MEASure:VTOP / 538  
:MEASure:WINDOW / 539  
:MEASure:XMAX / 540  
:MEASure:XMIN / 541

## 23 :MEASure Power Commands

:MEASure:ANGLE / 547  
:MEASure:APPARENT / 548  
:MEASure:CPLoss / 549  
:MEASure:CRESt / 550  
:MEASure:EFFiciency / 551  
:MEASure:ELOSSs / 552  
:MEASure:FACTOr / 553  
:MEASure:IPOWER / 554  
:MEASure:OFFTime / 555  
:MEASure:ONTime / 556  
:MEASure:OPOWер / 557  
:MEASure:PCURrent / 558  
:MEASure:PLOSSs / 559  
:MEASure:RDSon / 560  
:MEASure:REACTive / 561  
:MEASure:REAL / 562  
:MEASure:RIPPLE / 563  
:MEASure:TRESPonse / 564  
:MEASure:VCESat / 565

## 24 :MTEST Commands

:MTEST:ALL / 572  
:MTEST:AMASK:CREate / 573  
:MTEST:AMASK:SOURce / 574  
:MTEST:AMASK:UNITS / 575  
:MTEST:AMASK:XDELta / 576  
:MTEST:AMASK:YDELta / 577  
:MTEST:COUNT:FWAVEforms / 578  
:MTEST:COUNT:RESET / 579  
:MTEST:COUNT:TIME / 580  
:MTEST:COUNT:WAVEforms / 581  
:MTEST:DATA / 582  
:MTEST:DELETE / 583  
:MTEST:ENABLE / 584  
:MTEST:LOCK / 585  
:MTEST:RMODE / 586  
:MTEST:RMODE:FACTion:MEASure / 587  
:MTEST:RMODE:FACTion:PRINT / 588  
:MTEST:RMODE:FACTion:SAVE / 589  
:MTEST:RMODE:FACTion:STOP / 590

```
:MTEST:RMODE:SIGMa / 591
:MTEST:RMODE:TIME / 592
:MTEST:RMODE:WAVEforms / 593
:MTEST:SCALE:BIND / 594
:MTEST:SCALE:X1 / 595
:MTEST:SCALE:XDELta / 596
:MTEST:SCALE:Y1 / 597
:MTEST:SCALE:Y2 / 598
:MTEST:SOURce / 599
:MTEST:TITLe / 600
```

## 25 :POD Commands

```
:POD<n>:DISPlay / 603
:POD<n>:SIZE / 604
:POD<n>:THReShold / 605
```

## 26 :POWER Commands

```
:POWER:CLResponse:APPLy / 613
:POWER:CLResponse:FREQuency:STARt / 614
:POWER:CLResponse:FREQuency:STOP / 615
:POWER:CLResponse:VIEW / 616
:POWER:CLResponse:YMAXimum / 617
:POWER:CLResponse:YMINimum / 618
:POWER:DESKew / 619
:POWER:EFFiciency:APPLy / 620
:POWER:EFFiciency:TYPE / 621
:POWER:ENABLE / 622
:POWER:HARMonics:APPLy / 623
:POWER:HARMonics:DATA / 624
:POWER:HARMonics:DISPlay / 625
:POWER:HARMonics:FAILcount / 626
:POWER:HARMonics:LINE / 627
:POWER:HARMonics:POWERfactor / 628
:POWER:HARMonics:RPOWER / 629
:POWER:HARMonics:RPOWER:USER / 630
:POWER:HARMonics:RUNCount / 631
:POWER:HARMonics:STANDARD / 632
:POWER:HARMonics:STATus / 633
:POWER:HARMonics:THD / 634
:POWER:INRush:APPLy / 635
:POWER:INRush:EXIT / 636
```

:POWER:INRush:NEXT / 637  
:POWER:MODulation:APPLy / 638  
:POWER:MODulation:SOURce / 639  
:POWER:MODulation:TYPE / 640  
:POWER:ONOFF:APPLy / 641  
:POWER:ONOFF:EXIT / 642  
:POWER:ONOFF:NEXT / 643  
:POWER:ONOFF:TEST / 644  
:POWER:PSRR:APPLy / 645  
:POWER:PSRR:FREQuency:MAXimum / 646  
:POWER:PSRR:FREQuency:MINimum / 647  
:POWER:PSRR:RMAXimum / 648  
:POWER:QUALity:APPLy / 649  
:POWER:RIPPLE:APPLy / 650  
:POWER:SIGNals:AUTosetup / 651  
:POWER:SIGNals:CYCles:HARMonics / 652  
:POWER:SIGNals:CYCles:QUALity / 653  
:POWER:SIGNals:DURation:EFFiciency / 654  
:POWER:SIGNals:DURation:MODulation / 655  
:POWER:SIGNals:DURation:ONOFF:OFF / 656  
:POWER:SIGNals:DURation:ONOFF:ON / 657  
:POWER:SIGNals:DURation:RIPPLE / 658  
:POWER:SIGNals:DURation:TRANSient / 659  
:POWER:SIGNals:IEXPected / 660  
:POWER:SIGNals:OVERshoot / 661  
:POWER:SIGNals:VMAXimum:INRush / 662  
:POWER:SIGNals:VMAXimum:ONOFF:OFF / 663  
:POWER:SIGNals:VMAXimum:ONOFF:ON / 664  
:POWER:SIGNals:VSTeady:ONOFF:OFF / 665  
:POWER:SIGNals:VSTeady:ONOFF:ON / 666  
:POWER:SIGNals:VSTeady:TRANSient / 667  
:POWER:SIGNals:SOURce:CURREnt<i> / 668  
:POWER:SIGNals:SOURce:VOLTage<i> / 669  
:POWER:SLEW:APPLy / 670  
:POWER:SLEW:SOURce / 671  
:POWER:SWITch:APPLy / 672  
:POWER:SWITch:CONDuction / 673  
:POWER:SWITch:IREFerence / 674  
:POWER:SWITch:RDS / 675  
:POWER:SWITch:VCE / 676  
:POWER:SWITch:VREFerence / 677  
:POWER:TRANSient:APPLy / 678

```
:POWER:TRANSient:EXIT / 679  
:POWER:TRANSient:INITial / 680  
:POWER:TRANSient:INew / 681  
:POWER:TRANSient:NEXT / 682
```

## 27 :RECall Commands

```
:RECall:ARBitrary[:STARt] / 685  
:RECall:DBC[:STARt] / 686  
:RECall:FILEname / 687  
:RECall:LDF[:STARt] / 688  
:RECall:MASK[:STARt] / 689  
:RECall:PWD / 690  
:RECall:SETup[:STARt] / 691  
:RECall:WMEMory<r>[:STARt] / 692
```

## 28 :SAVE Commands

```
:SAVE:ARBitrary[:STARt] / 697  
:SAVE:FILEname / 698  
:SAVE:IMAGe[:STARt] / 699  
:SAVE:IMAGe:FACTors / 700  
:SAVE:IMAGe:FORMAT / 701  
:SAVE:IMAGe:INKSaver / 702  
:SAVE:IMAGe:PAlette / 703  
:SAVE:LISTER[:STARt] / 704  
:SAVE:MASK[:STARt] / 705  
:SAVE:MULTi[:STARt] / 706  
:SAVE:POWER[:STARt] / 707  
:SAVE:PWD / 708  
:SAVE:RESults[:STARt] / 709  
:SAVE:RESults:FORMAT:CURSor / 710  
:SAVE:RESults:FORMAT:MASK / 711  
:SAVE:RESults:FORMAT:MEASurement / 712  
:SAVE:RESults:FORMAT:SEARch / 713  
:SAVE:RESults:FORMAT:SEGmented / 714  
:SAVE[:SETup[:STARt]] / 715  
:SAVE:WAVEform[:STARt] / 716  
:SAVE:WAVEform:FORMAT / 717  
:SAVE:WAVEform:LENGTH / 718  
:SAVE:WAVEform:LENGTH:MAX / 719  
:SAVE:WAVEform:SEGmented / 720  
:SAVE:WMEMory:SOURce / 721
```

:SAVE:WMEMory[:STARt] / 722

## 29 :SBUS<n> Commands

General :SBUS<n> Commands / 725  
:SBUS<n>:DISPlay / 726  
:SBUS<n>:MODE / 727  
  
:SBUS<n>:A429 Commands / 728  
:SBUS<n>:A429:AUTosetup / 730  
:SBUS<n>:A429:BASE / 731  
:SBUS<n>:A429:COUNT:ERRor / 732  
:SBUS<n>:A429:COUNT:RESet / 733  
:SBUS<n>:A429:COUNT:WORD / 734  
:SBUS<n>:A429:FORMat / 735  
:SBUS<n>:A429:SIGNal / 736  
:SBUS<n>:A429:SOURce / 737  
:SBUS<n>:A429:SPeed / 738  
:SBUS<n>:A429:TRIGger:LABEL / 739  
:SBUS<n>:A429:TRIGger:PATTern:DATA / 740  
:SBUS<n>:A429:TRIGger:PATTern:SDI / 741  
:SBUS<n>:A429:TRIGger:PATTern:SSM / 742  
:SBUS<n>:A429:TRIGger:RANGE / 743  
:SBUS<n>:A429:TRIGger:TYPE / 744  
  
:SBUS<n>:CAN Commands / 745  
:SBUS<n>:CAN:COUNT:ERRor / 748  
:SBUS<n>:CAN:COUNT:OVERload / 749  
:SBUS<n>:CAN:COUNT:RESet / 750  
:SBUS<n>:CAN:COUNT:SPEC / 751  
:SBUS<n>:CAN:COUNT:TOTal / 752  
:SBUS<n>:CAN:COUNT:UTILization / 753  
:SBUS<n>:CAN:DISPlay / 754  
:SBUS<n>:CAN:FDSPoint / 755  
:SBUS<n>:CAN:FDSTandard / 756  
:SBUS<n>:CAN:SAMPLEpoint / 757  
:SBUS<n>:CAN:SIGNal:BAUDrate / 758  
:SBUS<n>:CAN:SIGNal:DEFinition / 759  
:SBUS<n>:CAN:SIGNal:FDBaudrate / 760  
:SBUS<n>:CAN:SOURce / 761  
:SBUS<n>:CAN:TRIGger / 762  
:SBUS<n>:CAN:TRIGger:IDFilter / 765  
:SBUS<n>:CAN:TRIGger:PATTern:DATA / 766

:SBUS<n>:CAN:TRIGger:PATTern:DATA:DLC / 767  
:SBUS<n>:CAN:TRIGger:PATTern:DATA:LENGth / 768  
:SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt / 769  
:SBUS<n>:CAN:TRIGger:PATTern:ID / 770  
:SBUS<n>:CAN:TRIGger:PATTern:ID:MODE / 771  
:SBUS<n>:CAN:TRIGger:SYMBolic:MESSage / 772  
:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNal / 773  
:SBUS<n>:CAN:TRIGger:SYMBolic:VALue / 774

:SBUS<n>:FLEXray Commands / 775  
:SBUS<n>:FLEXray:AUTosetup / 777  
:SBUS<n>:FLEXray:BAUDrate / 778  
:SBUS<n>:FLEXray:CHANnel / 779  
:SBUS<n>:FLEXray:COUNT:NULL / 780  
:SBUS<n>:FLEXray:COUNT:RESet / 781  
:SBUS<n>:FLEXray:COUNT:SYNC / 782  
:SBUS<n>:FLEXray:COUNT:TOTal / 783  
:SBUS<n>:FLEXray:SOURce / 784  
:SBUS<n>:FLEXray:TRIGger / 785  
:SBUS<n>:FLEXray:TRIGger:ERRor:TYPE / 786  
:SBUS<n>:FLEXray:TRIGger:EVENT:AUToset / 787  
:SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID / 788  
:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE / 789  
:SBUS<n>:FLEXray:TRIGger:FRAME:CCBase / 790  
:SBUS<n>:FLEXray:TRIGger:FRAME:CCRepetition / 791  
:SBUS<n>:FLEXray:TRIGger:FRAME:ID / 792  
:SBUS<n>:FLEXray:TRIGger:FRAME:TYPE / 793

:SBUS<n>:I2S Commands / 794  
:SBUS<n>:I2S:ALIGNment / 796  
:SBUS<n>:I2S:BASE / 797  
:SBUS<n>:I2S:CLOCK:SLOPe / 798  
:SBUS<n>:I2S:RWIDth / 799  
:SBUS<n>:I2S:SOURce:CLOCK / 800  
:SBUS<n>:I2S:SOURce:DATA / 801  
:SBUS<n>:I2S:SOURce:WSElect / 802  
:SBUS<n>:I2S:TRIGger / 803  
:SBUS<n>:I2S:TRIGger:AUDIO / 805  
:SBUS<n>:I2S:TRIGger:PATTern:DATA / 806  
:SBUS<n>:I2S:TRIGger:PATTern:FORMAT / 808  
:SBUS<n>:I2S:TRIGger:RANGE / 809  
:SBUS<n>:I2S:TWIDth / 811  
:SBUS<n>:I2S:WSLow / 812

:SBUS<n>:IIC Commands / 813  
:SBUS<n>:IIC:ASIZe / 814  
:SBUS<n>:IIC[:SOURce]:CLOCK / 815  
:SBUS<n>:IIC[:SOURce]:DATA / 816  
:SBUS<n>:IIC:TRIGger:PATTern:ADDResS / 817  
:SBUS<n>:IIC:TRIGger:PATTern:DATA / 818  
:SBUS<n>:IIC:TRIGger:PATTern:DATA2 / 819  
:SBUS<n>:IIC:TRIGger:QUALifier / 820  
:SBUS<n>:IIC:TRIGger[:TYPE] / 821

:SBUS<n>:LIN Commands / 823  
:SBUS<n>:LIN:DISPlay / 825  
:SBUS<n>:LIN:PARity / 826  
:SBUS<n>:LIN:SAMPlepoint / 827  
:SBUS<n>:LIN:SIGNAl:BAUDrate / 828  
:SBUS<n>:LIN:SOURce / 829  
:SBUS<n>:LIN:STANDARD / 830  
:SBUS<n>:LIN:SYNCbreak / 831  
:SBUS<n>:LIN:TRIGger / 832  
:SBUS<n>:LIN:TRIGger:ID / 833  
:SBUS<n>:LIN:TRIGger:PATTern:DATA / 834  
:SBUS<n>:LIN:TRIGger:PATTern:DATA:LENGTH / 836  
:SBUS<n>:LIN:TRIGger:PATTern:FORMAT / 837  
:SBUS<n>:LIN:TRIGger:SYMBOLic:FRAMe / 838  
:SBUS<n>:LIN:TRIGger:SYMBOLic:SIGNal / 839  
:SBUS<n>:LIN:TRIGger:SYMBOLic:VALue / 840

:SBUS<n>:M1553 Commands / 841  
:SBUS<n>:M1553:AUTosetup / 842  
:SBUS<n>:M1553:BASE / 843  
:SBUS<n>:M1553:SOURce / 844  
:SBUS<n>:M1553:TRIGger:PATTern:DATA / 845  
:SBUS<n>:M1553:TRIGger:RTA / 846  
:SBUS<n>:M1553:TRIGger:TYPE / 847

:SBUS<n>:SENT Commands / 848  
:SBUS<n>:SENT:CLOCK / 851  
:SBUS<n>:SENT:CRC / 852  
:SBUS<n>:SENT:DISPlay / 853  
:SBUS<n>:SENT:FORMAT / 855  
:SBUS<n>:SENT:IDLE / 857  
:SBUS<n>:SENT:LENGTH / 858  
:SBUS<n>:SENT:PPULse / 859

:SBUS<n>:SENT:SIGNAl<s>:DISPlay / 860  
:SBUS<n>:SENT:SIGNAl<s>:LENGth / 861  
:SBUS<n>:SENT:SIGNAl<s>:MULTiplier / 863  
:SBUS<n>:SENT:SIGNAl<s>:OFFSet / 864  
:SBUS<n>:SENT:SIGNAl<s>:ORDer / 865  
:SBUS<n>:SENT:SIGNAl<s>:STARt / 867  
:SBUS<n>:SENT:SOURce / 869  
:SBUS<n>:SENT:TOLerance / 871  
:SBUS<n>:SENT:TRIGger / 872  
:SBUS<n>:SENT:TRIGger:FAST:DATA / 874  
:SBUS<n>:SENT:TRIGger:SLOW:DATA / 875  
:SBUS<n>:SENT:TRIGger:SLOW:ID / 877  
:SBUS<n>:SENT:TRIGger:SLOW:ILENgh / 879  
:SBUS<n>:SENT:TRIGger:TOLERance / 880

:SBUS<n>:SPI Commands / 881  
:SBUS<n>:SPI:BITorder / 883  
:SBUS<n>:SPI:CLOCK:SLOPe / 884  
:SBUS<n>:SPI:CLOCK:TIMEout / 885  
:SBUS<n>:SPI:FRAMing / 886  
:SBUS<n>:SPI:SOURce:CLOCK / 887  
:SBUS<n>:SPI:SOURce:FRAMe / 888  
:SBUS<n>:SPI:SOURce:MISO / 889  
:SBUS<n>:SPI:SOURce:MOSI / 890  
:SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA / 891  
:SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh / 892  
:SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA / 893  
:SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh / 894  
:SBUS<n>:SPI:TRIGger:TYPE / 895  
:SBUS<n>:SPI:WIDTh / 896

:SBUS<n>:UART Commands / 897  
:SBUS<n>:UART:BASE / 900  
:SBUS<n>:UART:BAUDrate / 901  
:SBUS<n>:UART:BITorder / 902  
:SBUS<n>:UART:COUNT:ERRor / 903  
:SBUS<n>:UART:COUNT:RESet / 904  
:SBUS<n>:UART:COUNT:RXFRAMES / 905  
:SBUS<n>:UART:COUNT:TXFRAMES / 906  
:SBUS<n>:UART:FRAMing / 907  
:SBUS<n>:UART:PARity / 908  
:SBUS<n>:UART:POLarity / 909  
:SBUS<n>:UART:SOURce:RX / 910

:SBUS<n>:UART:SOURce:TX / 911  
:SBUS<n>:UART:TRIGger:BASE / 912  
:SBUS<n>:UART:TRIGger:BURSt / 913  
:SBUS<n>:UART:TRIGger:DATA / 914  
:SBUS<n>:UART:TRIGger:IDLE / 915  
:SBUS<n>:UART:TRIGger:QUALifier / 916  
:SBUS<n>:UART:TRIGger:TYPE / 917  
:SBUS<n>:UART:WIDTh / 918

## 30 :SEARch Commands

General :SEARch Commands / 920  
:SEARch:COUNt / 921  
:SEARch:EVENT / 922  
:SEARch:MODE / 923  
:SEARch:STATe / 924  
:SEARch:EDGE Commands / 925  
:SEARch:EDGE:SLOPe / 926  
:SEARch:EDGE:SOURce / 927  
:SEARch:GLITch Commands / 928  
:SEARch:GLITch:GREaterthan / 929  
:SEARch:GLITch:LESSthan / 930  
:SEARch:GLITch:POLarity / 931  
:SEARch:GLITch:QUALifier / 932  
:SEARch:GLITch:RANGe / 933  
:SEARch:GLITch:SOURce / 934  
:SEARch:PEAK Commands / 935  
:SEARch:PEAK:EXcursion / 936  
:SEARch:PEAK:NPEaks / 937  
:SEARch:PEAK:SOURce / 938  
:SEARch:PEAK:THreshold / 939  
:SEARch:RUNT Commands / 940  
:SEARch:RUNT:POLarity / 941  
:SEARch:RUNT:QUALifier / 942  
:SEARch:RUNT:SOURce / 943  
:SEARch:RUNT:TIME / 944  
:SEARch:TRANSition Commands / 945  
:SEARch:TRANSition:QUALifier / 946  
:SEARch:TRANSition:SLOPe / 947  
:SEARch:TRANSition:SOURce / 948

:SEARch:TRANSition:TIME / 949  
:SEARch:SERial:A429 Commands / 950  
    :SEARch:SERial:A429:LABel / 951  
    :SEARch:SERial:A429:MODE / 952  
    :SEARch:SERial:A429:PATTERn:DATA / 953  
    :SEARch:SERial:A429:PATTERn:SDI / 954  
    :SEARch:SERial:A429:PATTERn:SSM / 955  
  
:SEARch:SERial:CAN Commands / 956  
    :SEARch:SERial:CAN:MODE / 957  
    :SEARch:SERial:CAN:PATTERn:DATA / 959  
    :SEARch:SERial:CAN:PATTERn:DATA:LENGth / 960  
    :SEARch:SERial:CAN:PATTERn:ID / 961  
    :SEARch:SERial:CAN:PATTERn:ID:MODE / 962  
    :SEARch:SERial:CAN:SYMBOLic:MESSAge / 963  
    :SEARch:SERial:CAN:SYMBOLic:SIGNAl / 964  
    :SEARch:SERial:CAN:SYMBOLic:VALUe / 965  
  
:SEARch:SERial:FLEXray Commands / 966  
    :SEARch:SERial:FLEXray:CYCLE / 967  
    :SEARch:SERial:FLEXray:DATA / 968  
    :SEARch:SERial:FLEXray:DATA:LENGth / 969  
    :SEARch:SERial:FLEXray:FRAMe / 970  
    :SEARch:SERial:FLEXray:MODE / 971  
  
:SEARch:SERial:I2S Commands / 972  
    :SEARch:SERial:I2S:AUDio / 973  
    :SEARch:SERial:I2S:MODE / 974  
    :SEARch:SERial:I2S:PATTERn:DATA / 975  
    :SEARch:SERial:I2S:PATTERn:FORMAT / 976  
    :SEARch:SERial:I2S:RANGE / 977  
  
:SEARch:SERial:IIC Commands / 978  
    :SEARch:SERial:IIC:MODE / 979  
    :SEARch:SERial:IIC:PATTERn:ADDRess / 981  
    :SEARch:SERial:IIC:PATTERn:DATA / 982  
    :SEARch:SERial:IIC:PATTERn:DATA2 / 983  
    :SEARch:SERial:IIC:QUALifier / 984  
  
:SEARch:SERial:LIN Commands / 985  
    :SEARch:SERial:LIN:ID / 986  
    :SEARch:SERial:LIN:MODE / 987  
    :SEARch:SERial:LIN:PATTERn:DATA / 988  
    :SEARch:SERial:LIN:PATTERn:DATA:LENGth / 989

```
:SEARch:SERial:LIN:PATTERn:FORMAT / 990
:SEARch:SERial:LIN:SYMBolic:FRAMe / 991
:SEARch:SERial:LIN:SYMBolic:SIGNal / 992
:SEARch:SERial:LIN:SYMBolic:VALue / 993

:SEARch:SERial:M1553 Commands / 994
:SEARch:SERial:M1553:MODE / 995
:SEARch:SERial:M1553:PATTERn:DATA / 996
:SEARch:SERial:M1553:RTA / 997

:SEARch:SERial:SENT Commands / 998
:SEARch:SERial:SENT:FAST:DATA / 999
:SEARch:SERial:SENT:MODE / 1000
:SEARch:SERial:SENT:SLOW:DATA / 1001
:SEARch:SERial:SENT:SLOW:ID / 1002

:SEARch:SERial:SPI Commands / 1003
:SEARch:SERial:SPI:MODE / 1004
:SEARch:SERial:SPI:PATTERn:DATA / 1005
:SEARch:SERial:SPI:PATTERn:WIDTh / 1006

:SEARch:SERial:UART Commands / 1007
:SEARch:SERial:UART:DATA / 1008
:SEARch:SERial:UART:MODE / 1009
:SEARch:SERial:UART:QUALifier / 1010
```

### 31 :SYSTem Commands

```
:SYSTem:DATE / 1013
:SYSTem:DSP / 1014
:SYSTem:ERRor / 1015
:SYSTem:LOCK / 1016
:SYSTem:PERSONa[:MANufacturer] / 1017
:SYSTem:PERSONa[:MANufacturer]:DEFault / 1018
:SYSTem:PRESet / 1019
:SYSTem:PROTection:LOCK / 1022
:SYSTem:RLOGger / 1023
:SYSTem:RLOGger:DESTination / 1024
:SYSTem:RLOGger:DISPlay / 1025
:SYSTem:RLOGger:FNAME / 1026
:SYSTem:RLOGger:STATe / 1027
:SYSTem:RLOGger:TRANsparent / 1028
:SYSTem:RLOGger:WMODe / 1029
:SYSTem:SETup / 1030
:SYSTem:TIME / 1032
```

:SYSTem:TOUCH / 1033

## 32 :TIMEbase Commands

:TIMEbase:MODE / 1037  
:TIMEbase:POSIon / 1038  
:TIMEbase:RANGE / 1039  
:TIMEbase:REFerence / 1040  
:TIMEbase:SCALe / 1041  
:TIMEbase:VERNier / 1042  
:TIMEbase:WINDOW:POSIon / 1043  
:TIMEbase:WINDOW:RANGE / 1044  
:TIMEbase:WINDOW:SCALe / 1045

## 33 :TRIGger Commands

General :TRIGger Commands / 1049  
:TRIGger:FORCe / 1050  
:TRIGger:HFReject / 1051  
:TRIGger:HOLDoff / 1052  
:TRIGger:LEVel:ASETup / 1053  
:TRIGger:LEVel:HIGH / 1054  
:TRIGger:LEVel:LOW / 1055  
:TRIGger:MODE / 1056  
:TRIGger:NREject / 1057  
:TRIGger:SWEep / 1058  
  
:TRIGger:DELay Commands / 1059  
:TRIGger:DELay:ARM:SLOPe / 1060  
:TRIGger:DELay:ARM:SOURce / 1061  
:TRIGger:DELay:TDELay:TIME / 1062  
:TRIGger:DELay:TRIGger:COUNt / 1063  
:TRIGger:DELay:TRIGger:SLOPe / 1064  
:TRIGger:DELay:TRIGger:SOURce / 1065  
  
:TRIGger:EBURst Commands / 1066  
:TRIGger:EBURst:COUNt / 1067  
:TRIGger:EBURst:IDLE / 1068  
:TRIGger:EBURst:SLOPe / 1069  
:TRIGger:EBURst:SOURce / 1070  
  
:TRIGger[:EDGE] Commands / 1071  
:TRIGger[:EDGE]:COUpling / 1072  
:TRIGger[:EDGE]:LEVel / 1073  
:TRIGger[:EDGE]:REJect / 1074

:TRIGger[:EDGE]:SLOPe / 1075  
:TRIGger[:EDGE]:SOURce / 1076

:TRIGger:GLITch Commands / 1077  
:TRIGger:GLITch:GREaterthan / 1079  
:TRIGger:GLITch:LESSthan / 1080  
:TRIGger:GLITch:LEVel / 1081  
:TRIGger:GLITch:POLarity / 1082  
:TRIGger:GLITch:QUALifier / 1083  
:TRIGger:GLITch:RANGE / 1084  
:TRIGger:GLITch:SOURce / 1085

:TRIGger:NFC Commands / 1086  
:TRIGger:NFC:AEvent / 1087  
:TRIGger:NFC:ATTime / 1088  
:TRIGger:NFC:SOURce / 1089  
:TRIGger:NFC:STANDARD / 1090  
:TRIGger:NFC:TEVent / 1091  
:TRIGger:NFC:TIMEout / 1093  
:TRIGger:NFC:TIMEout:ENABLE / 1094  
:TRIGger:NFC:TIMEout:TIME / 1095

:TRIGger:OR Commands / 1096  
:TRIGger:OR / 1097

:TRIGger:PATTERn Commands / 1098  
:TRIGger:PATTERn / 1099  
:TRIGger:PATTERn:FORMat / 1101  
:TRIGger:PATTERn:GREaterthan / 1102  
:TRIGger:PATTERn:LESSthan / 1103  
:TRIGger:PATTERn:QUALifier / 1104  
:TRIGger:PATTERn:RANGE / 1105

:TRIGger:RUNT Commands / 1106  
:TRIGger:RUNT:POLarity / 1107  
:TRIGger:RUNT:QUALifier / 1108  
:TRIGger:RUNT:SOURce / 1109  
:TRIGger:RUNT:TIME / 1110

:TRIGger:SHOLD Commands / 1111  
:TRIGger:SHOLD:SLOPe / 1112  
:TRIGger:SHOLD:SOURce:CLOCK / 1113  
:TRIGger:SHOLD:SOURce:DATA / 1114  
:TRIGger:SHOLD:TIME:HOLD / 1115  
:TRIGger:SHOLD:TIME:SETup / 1116

```
:TRIGger:TRANSition Commands / 1117
:TRIGger:TRANSition:QUALifier / 1118
:TRIGger:TRANSition:SLOPe / 1119
:TRIGger:TRANSition:SOURce / 1120
:TRIGger:TRANSition:TIME / 1121

:TRIGger:TV Commands / 1122
:TRIGger:TV:LINE / 1123
:TRIGger:TV:MODE / 1124
:TRIGger:TV:POLarity / 1125
:TRIGger:TV:SOURce / 1126
:TRIGger:TV:STANDARD / 1127
:TRIGger:TV:UDTV:ENUMber / 1128
:TRIGger:TV:UDTV:HSync / 1129
:TRIGger:TV:UDTV:HTIMe / 1130
:TRIGger:TV:UDTV:PGTHan / 1131

:TRIGger:USB Commands / 1132
:TRIGger:USB:SOURce:DMINus / 1133
:TRIGger:USB:SOURce:DPLus / 1134
:TRIGger:USB:SPEEd / 1135
:TRIGger:USB:TRIGger / 1136

:TRIGger:ZONE Commands / 1137
:TRIGger:ZONE:SOURce / 1138
:TRIGger:ZONE:STATe / 1139
:TRIGger:ZONE<n>:MODE / 1140
:TRIGger:ZONE<n>:PLACement / 1141
:TRIGger:ZONE<n>:VALidity / 1142
:TRIGger:ZONE<n>:STATe / 1143
```

### 34 :WAVEform Commands

```
:WAVEform:BYTeorder / 1153
:WAVEform:COUNT / 1154
:WAVEform:DATA / 1155
:WAVEform:FORMat / 1157
:WAVEform:POINTs / 1158
:WAVEform:POINTs:MODE / 1160
:WAVEform:PREamble / 1162
:WAVEform:SEGmented:COUNt / 1165
:WAVEform:SEGmented:TTAG / 1166
:WAVEform:SOURce / 1167
:WAVEform:SOURce:SUBSource / 1171
```

```
:WAVEform:TYPE / 1172
:WAVEform:UNSIGNED / 1173
:WAVEform:VIEW / 1174
:WAVEform:XINCrement / 1175
:WAVEform:XORigin / 1176
:WAVEform:XREFerence / 1177
:WAVEform:YINCrement / 1178
:WAVEform:YORigin / 1179
:WAVEform:YREFerence / 1180
```

### 35 :WGEN<w> Commands

```
:WGEN<w>:ARBitrary:BYTeorder / 1185
:WGEN<w>:ARBitrary:DATA / 1186
:WGEN<w>:ARBitrary:DATA:ATTRibute:POINTs / 1187
:WGEN<w>:ARBitrary:DATA:CLEAR / 1188
:WGEN<w>:ARBitrary:DATA:DAC / 1189
:WGEN<w>:ARBitrary:INTerpolate / 1190
:WGEN<w>:ARBitrary:STORe / 1191
:WGEN<w>:FREQuency / 1192
:WGEN<w>:FUNCTION / 1193
:WGEN<w>:FUNCTION:PULSe:WIDTH / 1197
:WGEN<w>:FUNCTION:RAMP:SYMMetry / 1198
:WGEN<w>:FUNCTION:SQUare:DCYCle / 1199
:WGEN<w>:MODulation:AM:DEPTH / 1200
:WGEN<w>:MODulation:AM:FREQuency / 1201
:WGEN<w>:MODulation:FM:DEViation / 1202
:WGEN<w>:MODulation:FM:FREQuency / 1203
:WGEN<w>:MODulation:FSKey:FREQuency / 1204
:WGEN<w>:MODulation:FSKey:RATE / 1205
:WGEN<w>:MODulation:FUNCTION / 1206
:WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry / 1207
:WGEN<w>:MODulation:NOISE / 1208
:WGEN<w>:MODulation:STATe / 1209
:WGEN<w>:MODulation:TYPE / 1210
:WGEN<w>:OUTPut / 1212
:WGEN<w>:OUTPut:LOAD / 1213
:WGEN<w>:OUTPut:MODE / 1214
:WGEN<w>:OUTPut:POLarity / 1215
:WGEN<w>:OUTPut:SINGle / 1216
:WGEN<w>:PERiod / 1217
:WGEN<w>:RST / 1218
```

```
:WGEN<w>:VOLTage / 1219  
:WGEN<w>:VOLTage:HIGH / 1220  
:WGEN<w>:VOLTage:LOW / 1221  
:WGEN<w>:VOLTage:OFFSet / 1222
```

## 36 :WMEMory<r> Commands

```
:WMEMory<r>:CLEar / 1225  
:WMEMory<r>:DISPlay / 1226  
:WMEMory<r>:LABel / 1227  
:WMEMory<r>:SAVE / 1228  
:WMEMory<r>:SKEW / 1229  
:WMEMory<r>:YOFFset / 1230  
:WMEMory<r>:YRANge / 1231  
:WMEMory<r>:YScale / 1232
```

## 37 Obsolete and Discontinued Commands

```
:CHANnel:ACTivity / 1240  
:CHANnel:LABel / 1241  
:CHANnel:THRehold / 1242  
:CHANnel2:SKEW / 1243  
:CHANnel<n>:INPut / 1244  
:CHANnel<n>:PMODe / 1245  
:DISPlay:CONNect / 1246  
:DISPlay:ORDer / 1247  
:ERASe / 1248  
:EXTernal:PMODE / 1249  
:FUNCTION:GOFT:OPERation / 1250  
:FUNCTION:GOFT:SOURce1 / 1251  
:FUNCTION:GOFT:SOURce2 / 1252  
:FUNCTION:SOURce / 1253  
:FUNCTION:VIEW / 1254  
:HARDcopy:DESTination / 1255  
:HARDcopy:FILEname / 1256  
:HARDcopy:GRAYscale / 1257  
:HARDcopy:IGColors / 1258  
:HARDcopy:PDRiver / 1259  
:MEASure:LOWER / 1260  
:MEASure:SCRatch / 1261  
:MEASure:TDELta / 1262  
:MEASure:THReholds / 1263  
:MEASure:TMAX / 1264
```

:MEASure:TMIN / 1265  
:MEASure:TSTArt / 1266  
:MEASure:TSTOp / 1267  
:MEASure:TVOLt / 1268  
:MEASure:UPPer / 1269  
:MEASure:VDELta / 1270  
:MEASure:VSTArt / 1271  
:MEASure:VSTOp / 1272  
:MTEST:AMASK:{SAVE | STORe} / 1273  
:MTEST:AVERage / 1274  
:MTEST:AVERage:COUNt / 1275  
:MTEST:LOAD / 1276  
:MTEST:RUMode / 1277  
:MTEST:RUMode:SOFailure / 1278  
:MTEST:{STARt | STOP} / 1279  
:MTEST:TRIGger:SOURce / 1280  
:PRInt? / 1281  
:SAVE:IMAGe:AREA / 1283  
:SBUS<n>:LIN:SIGNal:DEFinition / 1284  
:SBUS<n>:SPI:SOURce:DATA / 1285  
:SYSTem:MENU / 1286  
:TIMEbase:DELay / 1287  
:TRIGger:THRehold / 1288  
:TRIGger:TV:TVMode / 1289

## 38 Error Messages

## 39 Status Reporting

Status Reporting Data Structures / 1301  
Status Byte Register (STB) / 1304  
Service Request Enable Register (SRE) / 1306  
Trigger Event Register (TER) / 1307  
Output Queue / 1308  
Message Queue / 1309  
(Standard) Event Status Register (ESR) / 1310  
(Standard) Event Status Enable Register (ESE) / 1311  
Error Queue / 1312  
Operation Status Event Register (:OPERegister[:EVENT]) / 1313

Operation Status Condition Register (:OPERegister:CONDition) /	1314
Arm Event Register (AER) /	1315
Overload Event Register (:OVLRegister) /	1316
Hardware Event Event Register (:HWERegister[:EVENT]) /	1317
Hardware Event Condition Register (:HWERegister:CONDition) /	1318
Mask Test Event Event Register (:MTERegister[:EVENT]) /	1319
Clearing Registers and Queues /	1320
Status Reporting Decision Chart /	1321

## 40 Synchronizing Acquisitions

Synchronization in the Programming Flow /	1324
Set Up the Oscilloscope /	1324
Acquire a Waveform /	1324
Retrieve Results /	1324
Blocking Synchronization /	1325
Polling Synchronization With Timeout /	1326
Synchronizing with a Single-Shot Device Under Test (DUT) /	1328
Synchronization with an Averaging Acquisition /	1330

## 41 More About Oscilloscope Commands

Command Classifications /	1334
Core Commands /	1334
Non-Core Commands /	1334
Obsolete Commands /	1334
Valid Command/Query Strings /	1335
Program Message Syntax /	1335
Duplicate Mnemonics /	1339
Tree Traversal Rules and Multiple Commands /	1339
Query Return Values /	1341
All Oscilloscope Commands Are Sequential /	1342

## 42 Programming Examples

- VISA COM Examples / 1344
  - VISA COM Example in Visual Basic / 1344
  - VISA COM Example in C# / 1353
  - VISA COM Example in Visual Basic .NET / 1362
  - VISA COM Example in Python / 1370
- VISA Examples / 1377
  - VISA Example in C / 1377
  - VISA Example in Visual Basic / 1386
  - VISA Example in C# / 1396
  - VISA Example in Visual Basic .NET / 1407
  - VISA Example in Python (PyVISA 1.5 and older) / 1417
  - VISA Example in Python (PyVISA 1.6 and newer) / 1423
- SICL Examples / 1430
  - SICL Example in C / 1430
  - SICL Example in Visual Basic / 1439
- SCPI.NET Examples / 1450
  - SCPI.NET Example in C# / 1450
  - SCPI.NET Example in Visual Basic .NET / 1456
  - SCPI.NET Example in IronPython / 1462

## Index

# 1 What's New

- What's New in Version 4.07 / 36
- What's New in Version 4.06 / 38
- What's New in Version 4.05 / 39
- Version 4.00 at Introduction / 41
- Command Differences From 4000 X-Series Oscilloscopes / 42

## What's New in Version 4.07

New features in version 4.07 of the InfiniiVision 3000T X-Series oscilloscope software are:

- Remote commands for remote command logging.
- Near Field Communication (NFC) trigger mode.

More detailed descriptions of the new and changed commands appear below.

### New Commands

Command	Description
:FFT:GATE (see <a href="#">page 369</a> )	Specifies whether the FFT is performed on the Main time base window or the Zoom window when the zoomed time base is displayed.
:FUNCTION<m>[:FFT]:GATE (see <a href="#">page 395</a> )	Specifies whether the FFT is performed on the Main time base window or the Zoom window when the zoomed time base is displayed.
:MARKer:X1:DISPlay (see <a href="#">page 444</a> )	Specifies whether the X1 cursor is displayed.
:MARKer:X2:DISPlay (see <a href="#">page 447</a> )	Specifies whether the X2 cursor is displayed.
:MARKer:Y1:DISPlay (see <a href="#">page 453</a> )	Specifies whether the Y1 cursor is displayed.
:MARKer:Y2:DISPlay (see <a href="#">page 455</a> )	Specifies whether the Y2 cursor is displayed.
:SYSTem:RLOGger (see <a href="#">page 1023</a> )	Enables or disables remote command logging, optionally specifying the log file name and write mode.
:SYSTem:RLOGger:DESTination (see <a href="#">page 1024</a> )	Specifies whether remote commands are logged to a text file (on a connected USB storage device), logged to the screen, or both.
:SYSTem:RLOGger:DISPlay (see <a href="#">page 1025</a> )	Enables or disables the screen display of logged remote commands and their return values (if applicable).
:SYSTem:RLOGger:FNAME (see <a href="#">page 1026</a> )	Specifies the remote command log file name.
:SYSTem:RLOGger:STATE (see <a href="#">page 1027</a> )	Enables or disables remote command logging.
:SYSTem:RLOGger:TRANsparen t (see <a href="#">page 1028</a> )	Specifies whether the screen display background for remote command logging is transparent or solid.
:SYSTem:RLOGger:WMODE (see <a href="#">page 1029</a> )	Specifies the remote command logging write mode (either CREate or APPend).
:TRIGger:NFC Commands (see <a href="#">page 1086</a> )	Commands for setting up the Near Field Communication (NFC) trigger mode.

**Changed Commands**

Command	Differences
:CALibrate:OUTPut (see page 271)	The NFC option becomes available in the Near Field Communication (NFC) trigger mode when the ATRigger (Arm & Trigger) trigger event is selected.
:FUNCTION<m>:OPERation (see page 405)	The MAXimum, MINimum, and PEAK operations are added.
:TRIGger:MODE (see page 1056)	The NFC option is now available for setting the Near Field Communication (NFC) trigger mode. See " <b>:TRIGger:NFC Commands</b> " on page 1086.

## What's New in Version 4.06

New features in version 4.06 of the InfiniiVision 3000T X-Series oscilloscope software are:

- The Control Loop Response (Bode) power analysis now lets you select a phase plot as well as a gain plot.
- The ISO standard for CAN FD is now supported.
- Up to 10 annotations are supported.

More detailed descriptions of the new and changed commands appear below.

### New Commands

Command	Description
:POWER:CLResponse:VIEW (see <a href="#">page 616</a> )	When the Control Loop Response (Bode) power analysis is selected, this command selects whether to display a gain or phase plot.
:SBUS<n>:CAN:FDSTandard (see <a href="#">page 756</a> )	Lets you pick the standard that will be used when decoding or triggering on FD frames, ISO, or non-ISO.

### Changed Commands

Command	Differences
:DISPLAY:ANNAnnotation<n> (see <a href="#">page 332</a> )	You can now define up to 10 annotations.
:DISPLAY:ANNAnnotation<n>:BACKGROUND (see <a href="#">page 333</a> )	
:DISPLAY:ANNAnnotation<n>:COLOR (see <a href="#">page 334</a> )	
:DISPLAY:ANNAnnotation<n>:TEXT (see <a href="#">page 335</a> )	
:DISPLAY:ANNAnnotation<n>:X1POSITION (see <a href="#">page 336</a> )	
:DISPLAY:ANNAnnotation<n>:Y1POSITION (see <a href="#">page 337</a> )	
:SBUS<n>:CAN:TRIGGER (see <a href="#">page 762</a> )	Has some description changes related to the CAN FD ISO support changes.

## What's New in Version 4.05

New features in version 4.05 of the InfiniiVision 3000T X-Series oscilloscope software are:

- Being able to load LIN symbolic data from an LDF (\*.ldf) file into the oscilloscope, display it in the decode, and use it to set up triggers and protocol decode searches.

More detailed descriptions of the new and changed commands appear below.

New Commands

Command	Description
:POWER:CLResponse:APPLy (see <a href="#">page 613</a> )	Performs the Control Loop Response (Bode) power analysis.
:POWER:CLResponse:FREQuency:STARt (see <a href="#">page 614</a> )	Sets the sweep start frequency value.
:POWER:CLResponse:FREQuency:STOP (see <a href="#">page 615</a> )	Sets the sweep stop frequency value.
:POWER:CLResponse:YMAXimum (see <a href="#">page 617</a> )	Specifies the Bode plot's initial vertical scale maximum value.
:POWER:CLResponse:YMINimum (see <a href="#">page 618</a> )	Specifies the Bode plot's initial vertical scale minimum value.
:POWER:HARMonics:RPOWer (see <a href="#">page 629</a> )	When Class D is selected as the current harmonics analysis standard, this command specifies whether the Real Power value used for the current-per-watt measurement is measured by the oscilloscope or is defined by the user.
:POWER:HARMonics:RPOWer:USER (see <a href="#">page 630</a> )	When Class D is selected as the current harmonics analysis standard and you have chosen to use a user-defined Real Power value, this command specifies the Real Power value used in the current-per-watt measurement.
:RECall:LDF[:STARt] (see <a href="#">page 688</a> )	Loads a LIN description file (LDF) into the oscilloscope.
:SBUS<n>:LIN:DISPlay (see <a href="#">page 825</a> )	Specifies whether symbolic values (from a loaded LDF file) or hexadecimal values are displayed in the decode waveform and the Lister window
:SBUS<n>:LIN:TRIGger:SYMBOLic:FRAMe (see <a href="#">page 838</a> )	Specifies the frame to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FRAMe or FSIGnal.
:SBUS<n>:LIN:TRIGger:SYMBOLic:SIGNal (see <a href="#">page 839</a> )	Specifies signal to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FSIGnal.

Command	Description
:SBUS<n>:LIN:TRIGger:SYMBOLic:VALue (see <a href="#">page 840</a> )	Specifies signal value to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FSIGnal.
:SEARch:SERial:LIN:SYMBOLic:FRAMe (see <a href="#">page 991</a> )	Specifies the message to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FRAMe or FSIGnal.
:SEARch:SERial:LIN:SYMBOLic:SIGNal (see <a href="#">page 992</a> )	Specifies signal to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FSIGnal.
:SEARch:SERial:LIN:SYMBOLic:VALue (see <a href="#">page 993</a> )	Specifies signal value to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FSIGnal.
:TRIGger:USB Commands (see <a href="#">page 1132</a> )	Resurrected from the 3000 X-Series oscilloscopes, these commands let you trigger on a Start of Packet (SOP), End of Packet (EOP), Reset Complete, Enter Suspend, or Exit Suspend signal on the differential USB data lines. USB Low Speed and Full Speed are supported by this trigger.
:WGEN<w>:OUTPut:MODE (see <a href="#">page 1214</a> )	Specifies whether the defined waveform is output continuously or as a single cycle (single-shot).
:WGEN<w>:OUTPut:SINGle (see <a href="#">page 1216</a> )	When the single-shot output mode is selected, this command causes a single cycle of the defined waveform to be output.

#### Changed Commands

Command	Differences
:SBUS<n>:LIN:TRIGger (see <a href="#">page 832</a> )	You can now trigger on symbolic frames (with parameter FRAMe) or frames and signal values (with parameter FSIGnal).
:SEARch:SERial:LIN:MODE (see <a href="#">page 987</a> )	You can now search for symbolic messages (with parameter FRAMe) or frames and signal values (with parameter FSIGnal).
:TRIGger:MODE (see <a href="#">page 1056</a> )	The parameter USB is now allowed to permit USB triggering.

## Version 4.00 at Introduction

The Keysight InfiniiVision 3000T X-Series oscilloscopes were introduced with version 4.00 of oscilloscope operating software.

The command set is most closely related to the InfiniiVision 4000 X-Series oscilloscopes (and the 3000 X-Series, 7000A/B Series, 6000 Series, and 54620/54640 Series oscilloscopes before them). For more information, see ["Command Differences From 4000 X-Series Oscilloscopes"](#) on page 42.

## Command Differences From 4000 X-Series Oscilloscopes

The Keysight InfiniiVision 3000T X-Series oscilloscopes command set is most closely related to the InfiniiVision 4000 X-Series oscilloscopes (and the 3000 X-Series, 7000A/B Series, 6000 Series, and 54620/54640 Series oscilloscopes before them).

The main differences between the version 4.00 programming command set for the InfiniiVision 3000T X-Series oscilloscopes and the 3.20 programming command set for the InfiniiVision 4000 X-Series oscilloscopes are related to:

- Dedicated FFT function (and selectable math functions – 4000 X-Series oscilloscopes have four selectable math functions).
- SENT serial decode and triggering option.
- Updates to support CAN FD serial decode and triggering.
- Counter feature (when DSOXDVMCTR option is licensed).
- New built-in demo signals (with Option EDU license that comes with the N6455A Education Kit).
- One waveform generator output (4000 X-Series oscilloscopes have two waveform generator outputs).
- No 10 MHz REF connector.
- No support for USB 2.0 serial decode, triggering, or signal quality analysis.

More detailed descriptions of the new, changed, obsolete, and discontinued commands appear below.

### New Commands

Command	Description
:CHANnel<n>:PROBe:MMODel (see <a href="#">page 292</a> )	Sets the model number of the supported Tektronix probe.
:COUNter Commands (see <a href="#">page 301</a> )	Commands for controlling the optional DSOXDVMCTR counter feature.
:DISPlay:ANNotation<n>:X1Position (see <a href="#">page 336</a> )	Sets the horizontal position of one of the four annotations.
:DISPlay:ANNotation<n>:Y1Position (see <a href="#">page 337</a> )	Sets the horizontal position of one of the four annotations.
:FFT Commands (see <a href="#">page 359</a> )	Commands for using the FFT function feature.
:FUNCtion<m>:CLEar (see <a href="#">page 390</a> )	When the :FUNCtion<m>:OPERation is AVERage, MAXHold, or MINHold, this command clears the number of evaluated waveforms.

Command	Description
:FUNCTION<m>[:FFT]:FREQuency:STARt (see <a href="#">page 393</a> )	Lets you set the displayed frequency range using start and stop frequency values.
:FUNCTION<m>[:FFT]:FREQuency:STOP (see <a href="#">page 394</a> )	
:FUNCTION<m>:SMOOth:POINTs (see <a href="#">page 412</a> )	Sets the number of smoothing points for the new SMOOTH :FUNCTION<m>:OPERation.
:MEASure:BRATe (see <a href="#">page 480</a> )	Installs a bit rate measurement on screen or returns the measured value.
:MEASure:RDSon (see <a href="#">page 560</a> )	Installs an RDS(on) power measurement on screen or returns the measured value.
:MEASure:VCESat (see <a href="#">page 565</a> )	Installs a VCE(sat) power measurement on screen or returns the measured value.
:POWER:EFFiciency:TYPE (see <a href="#">page 621</a> )	Specifies the type of power that is being converted from the input to the output.
:SAVE:RESults:[STARt] (see <a href="#">page 709</a> )	Saves analysis results to a comma-separated values (*.csv) file on a USB storage device.
:SAVE:RESults:FORMAT:CURSo r (see <a href="#">page 710</a> )	Specifies whether cursor values will be included when analysis results are saved.
:SAVE:RESults:FORMAT:MASK (see <a href="#">page 711</a> )	Specifies whether mask statistics will be included when analysis results are saved.
:SAVE:RESults:FORMAT:MEASu rement (see <a href="#">page 712</a> )	Specifies whether measurement results will be included when analysis results are saved.
:SAVE:RESults:FORMAT:SEARc h (see <a href="#">page 713</a> )	Specifies whether found search event times will be included when analysis results are saved.
:SAVE:RESults:FORMAT:SEGMe nted (see <a href="#">page 714</a> )	Specifies whether segmented memory acquisition times will be included when analysis results are saved.
:SEARch:PEAK Commands (see <a href="#">page 935</a> )	Commands to set up searching for FFT peaks.
:SBUS<n>:CAN:COUNT:SPEC (see <a href="#">page 751</a> )	Returns the count for the number of Spec errors (Ack + Form + Stuff + CRC errors).
:SBUS<n>:CAN:FDSPoint (see <a href="#">page 755</a> )	Sets the CAN FD sample point.
:SBUS<n>:CAN:SIGNal:FDBaud rate (see <a href="#">page 760</a> )	Sets the CAN FD baud rate.
:SBUS<n>:CAN:TRIGger:IDFilte r (see <a href="#">page 765</a> )	Specifies, in certain error and bit trigger modes, whether triggers are filtered by CAN IDs.
:SBUS<n>:CAN:TRIGger:PATTER n:DATA:DLC (see <a href="#">page 767</a> )	Specifies the DLC value to be used in the CAN FD data trigger mode.

Command	Description
:SBUS<n>:CAN:TRIGger:PATTer n:DATA:START (see <a href="#">page 769</a> )	Specifies the starting byte position for CAN FD data triggers.
:SBUS<n>:SENT Commands (see <a href="#">page 848</a> )	Commands for using the SENT triggering and serial decode feature.
:SEARch:SERial:SENT Commands (see <a href="#">page 998</a> )	Commands for searching SENT serial decode events.
:SYSTem:PERSONa[:MANufacturer] (see <a href="#">page 1017</a> )	Lets you change the manufacturer string portion of the identification string returned by the *IDN? query.
:SYSTem:PERSONa[:MANufacturer]:DEFault (see <a href="#">page 1018</a> )	Sets manufacturer string to "KEYSIGHT TECHNOLOGIES".

### Changed Commands

Command	Differences From InfiniiVision 4000 X-Series Oscilloscopes
:ACQuire:MODE (see <a href="#">page 247</a> )	The ETIMe option is not supported in the 3000T X-Series oscilloscopes.
:DEMO:FUNCTION (see <a href="#">page 314</a> )	The DCMotor, HARMonics, COUPLing, CFD, and SENT functions are now available with the DSOXEDK educator's kit license.
:DISPlay:ANNotation<n> (see <a href="#">page 332</a> )	You can now define up to four annotations.
:DISPlay:ANNotation<n>:BACKground (see <a href="#">page 333</a> )	
:DISPlay:ANNotation<n>:COLor (see <a href="#">page 334</a> )	
:DISPlay:ANNotation<n>:TEXT (see <a href="#">page 335</a> )	
:DISPlay:SIDebar (see <a href="#">page 344</a> )	The EVENTs and COUNter options are now available.
:DISPlay:VECTors (see <a href="#">page 346</a> )	Always ON (no display as dots option).
:DVM:MODE (see <a href="#">page 351</a> )	The FREQuency option has been replaced by the new <a href="#">Chapter 11</a> , “:COUNter Commands,” starting on page 301.
:EXTernal:RANGE (see <a href="#">page 356</a> )	When using 1:1 probe attenuation, the range can only be set to 8.0 V.
:FUNCTION<m>:OPERation (see <a href="#">page 405</a> )	The SMOOTH, ENvelope, MAXHold, and MINHold operations are added.
:POWER:SIGNals:AUTosetup (see <a href="#">page 651</a> )	The RDSVce option is now available.
:SBUS<n>:CAN:COUNT:OVERload (see <a href="#">page 749</a> )	Always returns zero.

Command	Differences From InfiniiVision 4000 X-Series Oscilloscopes
:SBUS<n>:CAN:TRIGger (see page 762)	Has additional parameters that support CAN FD triggering.
:SBUS<n>:MODE (see page 727)	The SENT mode is now available with the DSOX3SENSOR SENT serial decode and triggering license.
:SEARch:MODE (see page 923)	The PEAK option has been added to enable searching for FFT peaks (see “:SEARch:PEAK Commands” on page 935).
:SEARch:SERial:CAN:MODE (see page 957)	Has additional parameters that support CAN FD searching.
:WAVeform:SOURce:SUBSourc e (see page 1171)	You can use FAST (alias for SUB0) to get SENT fast channel data or SLOW (alias for SBUS1) to get SENT slow channel data.

### Discontinued Commands

Discontinued Command	Current Command Equivalent	Comments
:ACQuire:RSIGnal	none	There is no 10 MHz REF connector on the 3000T X-Series oscilloscopes.
:COMPliance Commands	none	USB signal quality analysis is not supported on the 3000T X-Series oscilloscopes.
:DVM:FREQuency?	:COUNter:CURREnt? (see page 303)	The :DVM:FREQuency? query has been replaced by the new Chapter 11, “:COUNter Commands,” starting on page 301.
:SAVE:COMPliance:USB[:STARt ]	none	USB signal quality analysis is not supported on the 3000T X-Series oscilloscopes.
:SBUS<n>:USB Commands	none	USB serial decode and triggering is not supported on the 3000T X-Series oscilloscopes.
:SEARch:SERial:USB Commands	none	Searching USB serial decode is not supported on the 3000T X-Series oscilloscopes.
:TIMEbase:REFClock	none	There is no 10 MHz REF connector on the 3000T X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:WGEN<w>:TRACk	none	There is only one waveform generator output on the 3000T X-Series oscilloscopes.
:WGEN<w>:TRACk:AMPLitude	none	
:WGEN<w>:TRACk:CSIGnal	none	
:WGEN<w>:TRACk:FREQuency	none	

## 2 Setting Up

Step 1. Install Keysight IO Libraries Suite software / 48

Step 2. Connect and set up the oscilloscope / 49

Step 3. Verify the oscilloscope connection / 51

This chapter explains how to install the Keysight IO Libraries Suite software, connect the oscilloscope to the controller PC, set up the oscilloscope, and verify the oscilloscope connection.

## Step 1. Install Keysight IO Libraries Suite software

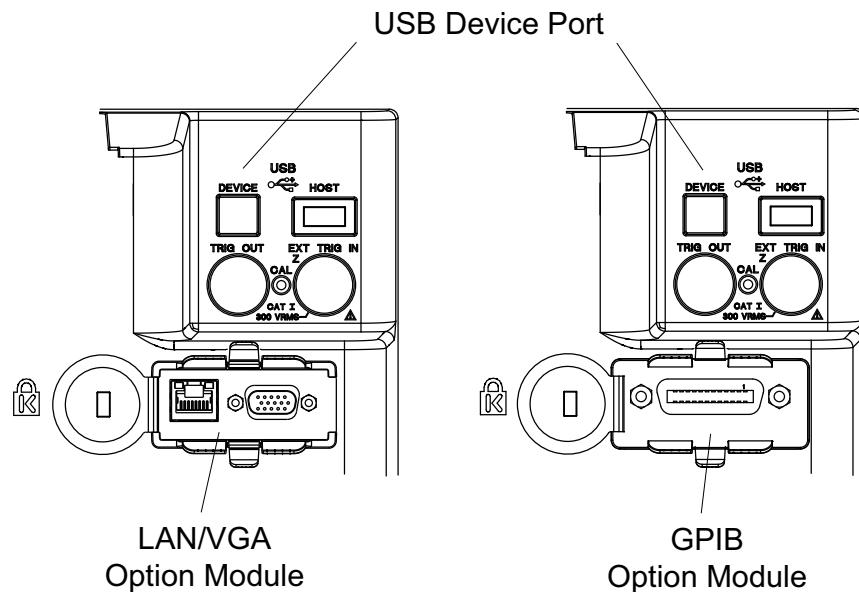
- 1** Download the Keysight IO Libraries Suite software from the Keysight web site at:
  - <http://www.keysight.com/find/iolib>
- 2** Run the setup file, and follow its installation instructions.

## Step 2. Connect and set up the oscilloscope

The 3000T X-Series oscilloscope has two different interfaces you can use for programming:

- USB (device port).
- LAN. To configure the LAN interface, press the **[Utility]** key on the front panel, then press the **I/O** softkey, then press the **Configure** softkey.

These interfaces are always active.



**Figure 1** Control Connectors on Rear Panel

### Using the USB (Device) Interface

- 1 Connect a USB cable from the controller PC's USB port to the "USB DEVICE" port on the back of the oscilloscope.

This is a USB 2.0 high-speed port.

### Using the LAN Interface

- 1 If the controller PC is not already connected to the local area network (LAN), do that first.
- 2 Contact your network administrator about adding the oscilloscope to the network.

Find out if automatic configuration via DHCP or AutoIP can be used. Also, find out whether your network supports Dynamic DNS or Multicast DNS.

If automatic configuration is not supported, get the oscilloscope's network parameters (hostname, domain, IP address, subnet mask, gateway IP, DNS IP, etc.).

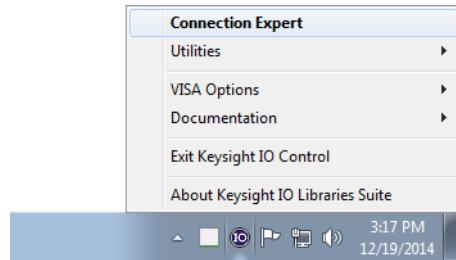
- 3 Connect the oscilloscope to the local area network (LAN) by inserting LAN cable into the "LAN" port on the LAN/VGA option module.
- 4 Configure the oscilloscope's LAN interface:
  - a Press the **Configure** softkey until "LAN" is selected.
  - b Press the **LAN Settings** softkey.
  - c Press the **Config** softkey, and enable all the configuration options supported by your network.
  - d If automatic configuration is not supported, press the **Addresses** softkey.  
Use the **Modify** softkey (and the other softkeys and the Entry knob) to enter the IP Address, Subnet Mask, Gateway IP, and DNS IP values.  
When you are done, press the **[Back up]** key.
  - e Press the **Host name** softkey. Use the softkeys and the Entry knob to enter the Host name.  
When you are done, press the **[Back up]** key.

## Using the GPIB Interface

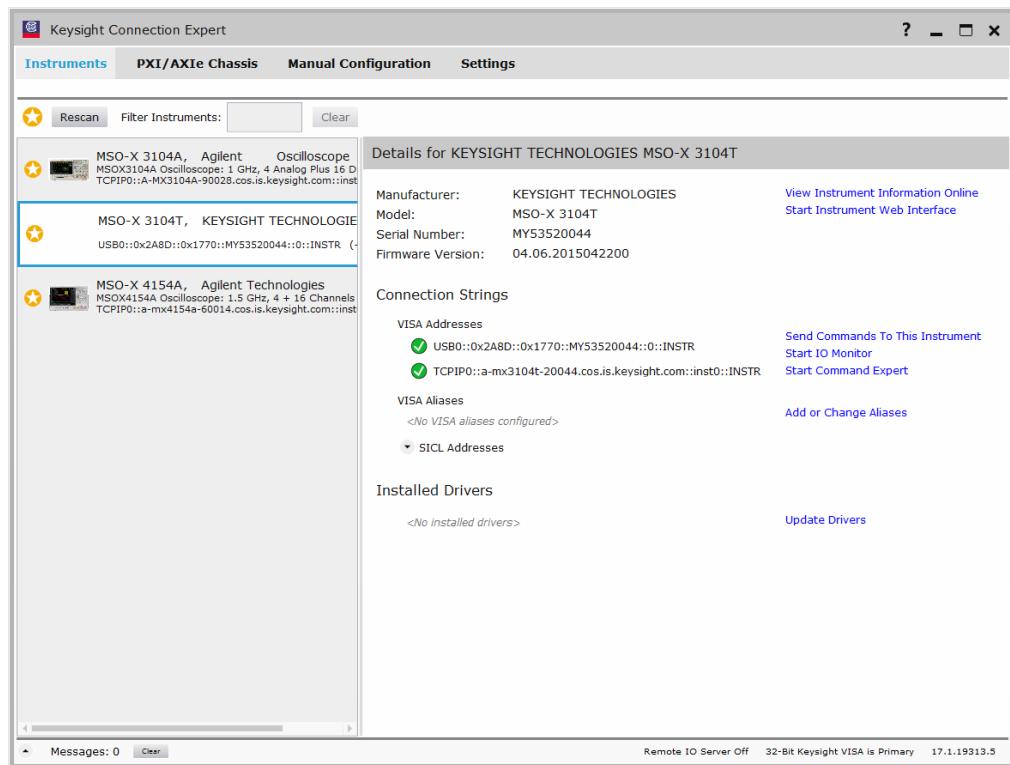
- 1 Connect a GPIB cable from the controller PC's GPIB interface to the "GPIB" port on the GPIB option module.
- 2 Configure the oscilloscope's GPIB interface:
  - a Press the **Configure** softkey until "GPIB" is selected.
  - b Use the Entry knob to select the **Address** value.

## Step 3. Verify the oscilloscope connection

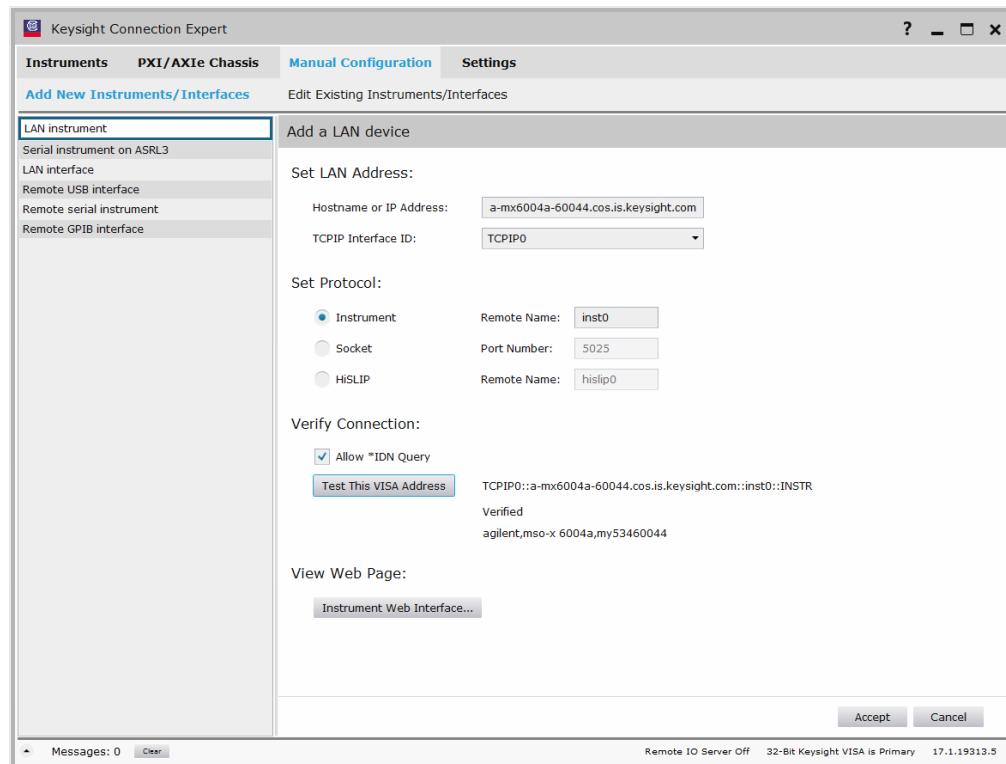
- 1 On the controller PC, click on the Keysight IO Control icon in the taskbar and choose **Connection Expert** from the popup menu.



- 2 In the Keysight Connection Expert application, instruments connected to the controller's USB and GPIB interfaces as well as instruments on the same LAN subnet should automatically appear in the Instruments tab.



- 3 If your instrument does not appear, you can add it using the Manual Configuration tab.

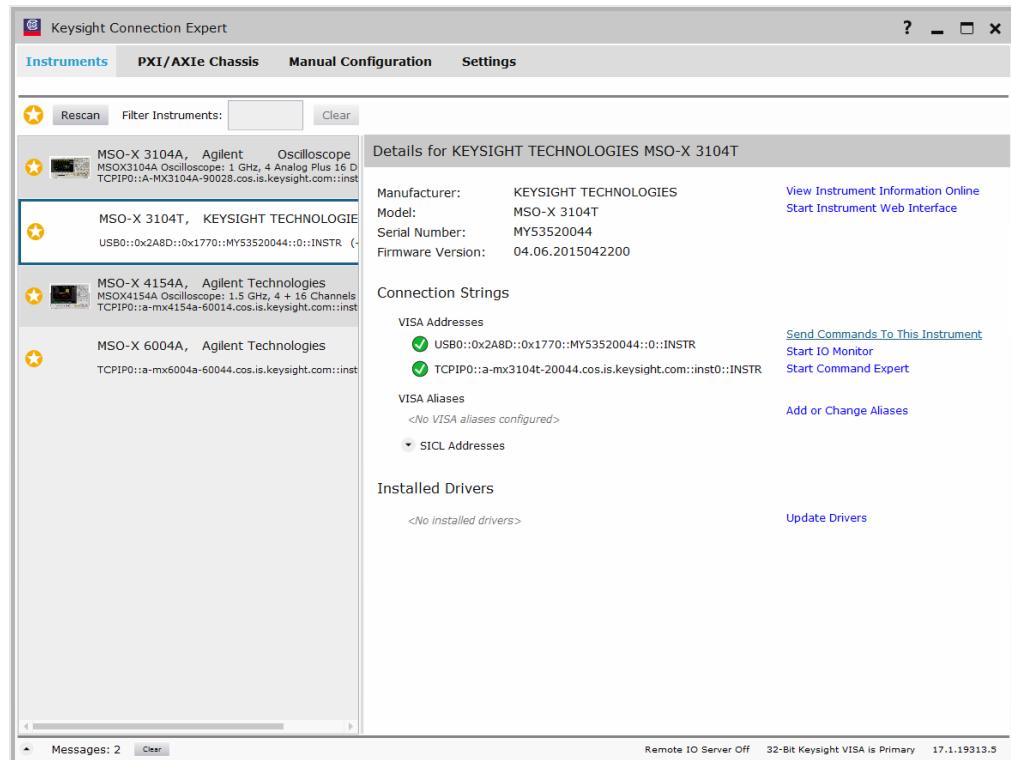


For example, to add a device:

- a Select **LAN instrument** in the list on the left.
- b Enter the oscilloscope's **Hostname or IP address**.
- c Select the protocol.
- d Select **Instrument** under Set Protocol.
- e Click **Test This VISA Address** to verify the connection.
- f If the connection test is successful, click **Accept** to add the instrument.

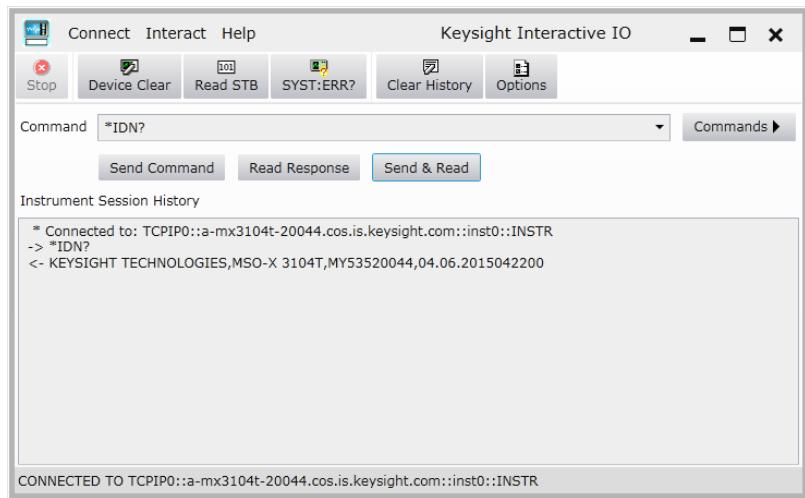
If the connection test is not successful, go back and verify the LAN connections and the oscilloscope setup.

- 4** Test some commands on the instrument:
- In the Details for the selected instrument, click **Send Commands To This Instrument**.



- In the Keysight Interactive IO application, enter commands in the **Command** field and press **Send Command**, **Read Response**, or **Send & Read**.

## 2 Setting Up



- c Choose **Connect > Exit** from the menu to exit the Keysight Interactive IO application.
- 5 In the Keysight Connection Expert application, choose **File > Exit** from the menu to exit the application.

# 3 Getting Started

Basic Oscilloscope Program Structure / 56

Programming the Oscilloscope / 58

Other Ways of Sending Commands / 67

This chapter gives you an overview of programming the 3000T X-Series oscilloscopes. It describes basic oscilloscope program structure and shows how to program the oscilloscope using a few simple examples.

The getting started examples show how to send oscilloscope setup, data capture, and query commands, and they show how to read query results.

**NOTE**

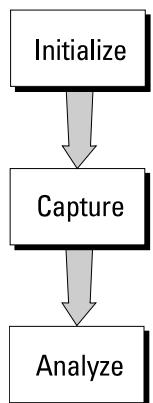
### Language for Program Examples

The programming examples in this guide are written in Visual Basic using the Keysight VISA COM library.

---

## Basic Oscilloscope Program Structure

The following figure shows the basic structure of every program you will write for the oscilloscope.



### Initializing

To ensure consistent, repeatable performance, you need to start the program, controller, and oscilloscope in a known state. Without correct initialization, your program may run correctly in one instance and not in another. This might be due to changes made in configuration by previous program runs or from the front panel of the oscilloscope.

- Program initialization defines and initializes variables, allocates memory, or tests system configuration.
- Controller initialization ensures that the interface to the oscilloscope is properly set up and ready for data transfer.
- Oscilloscope initialization sets the channel configuration, channel labels, threshold voltages, trigger specification, trigger mode, timebase, and acquisition type.

### Capturing Data

Once you initialize the oscilloscope, you can begin capturing data for analysis. Remember that while the oscilloscope is responding to commands from the controller, it is not performing acquisitions. Also, when you change the oscilloscope configuration, any data already captured will most likely be rendered.

To collect data, you use the :DIGitize command. This command clears the waveform buffers and starts the acquisition process. Acquisition continues until acquisition memory is full, then stops. The acquired data is displayed by the oscilloscope, and the captured data can be measured, stored in acquisition

memory in the oscilloscope, or transferred to the controller for further analysis. Any additional commands sent while :DIGITIZE is working are buffered until :DIGITIZE is complete.

You could also put the oscilloscope into run mode, then use a wait loop in your program to ensure that the oscilloscope has completed at least one acquisition before you make a measurement. Keysight does not recommend this because the needed length of the wait loop may vary, causing your program to fail. :DIGITIZE, on the other hand, ensures that data capture is complete. Also, :DIGITIZE, when complete, stops the acquisition process so that all measurements are on displayed data, not on a constantly changing data set.

## Analyzing Captured Data

After the oscilloscope has completed an acquisition, you can find out more about the data, either by using the oscilloscope measurements or by transferring the data to the controller for manipulation by your program. Built-in measurements include: frequency, duty cycle, period, positive pulse width, and negative pulse width.

Using the :WAVEFORM commands, you can transfer the data to your controller. You may want to display the data, compare it to a known good measurement, or simply check logic patterns at various time intervals in the acquisition.

## Programming the Oscilloscope

- "Referencing the IO Library" on page 58
- "Opening the Oscilloscope Connection via the IO Library" on page 59
- "Using :AUToscale to Automate Oscilloscope Setup" on page 60
- "Using Other Oscilloscope Setup Commands" on page 60
- "Capturing Data with the :DIGitize Command" on page 61
- "Reading Query Responses from the Oscilloscope" on page 63
- "Reading Query Results into String Variables" on page 64
- "Reading Query Results into Numeric Variables" on page 64
- "Reading Definite-Length Block Query Response Data" on page 64
- "Sending Multiple Queries and Reading Results" on page 65
- "Checking Instrument Status" on page 66

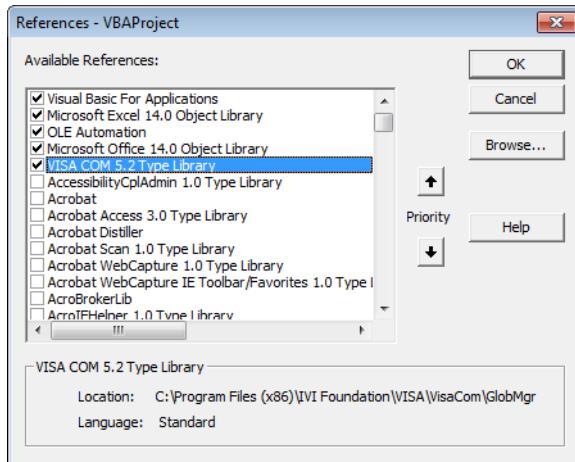
### Referencing the IO Library

No matter which instrument programming library you use (SICL, VISA, or VISA COM), you must reference the library from your program.

In C/C++, you must tell the compiler where to find the include and library files (see the Keysight IO Libraries Suite documentation for more information).

To reference the Keysight VISA COM library in Visual Basic for Applications (VBA, which comes with Microsoft Office products like Excel):

- 1 Choose **Tools > References...** from the main menu.
- 2 In the References dialog, check the "VISA COM 5.5 Type Library".



- 3 Click **OK**.

To reference the Keysight VISA COM library in Microsoft Visual Basic 6.0:

- 1** Choose **Project > References...** from the main menu.
- 2** In the References dialog, check the "VISA COM 5.5 Type Library".
- 3** Click **OK**.

## Opening the Oscilloscope Connection via the IO Library

PC controllers communicate with the oscilloscope by sending and receiving messages over a remote interface. Once you have opened a connection to the oscilloscope over the remote interface, programming instructions normally appear as ASCII character strings embedded inside write statements of the programming language. Read statements are used to read query responses from the oscilloscope.

For example, when using the Keysight VISA COM library in Visual Basic (after opening the connection to the instrument using the ResourceManager object's Open method), the FormattedIO488 object's WriteString, WriteNumber, WriteList, or WriteIEEEBlock methods are used for sending commands and queries. After a query is sent, the response is read using the ReadString, ReadNumber, ReadList, or ReadIEEEBlock methods.

The following Visual Basic statements open the connection and send a command that turns on the oscilloscope's label display.

```
Dim myMgr As VisaComLib.ResourceManager
Dim myScope As VisaComLib.FormattedIO488

Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488

' Open the connection to the oscilloscope. Get the VISA Address from the
' Keysight Connection Expert (installed with Keysight IO Libraries Suite
').
Set myScope.IO = myMgr.Open("<VISA Address>")

' Send a command.
myScope.WriteString ":DISPLAY:LABEL ON"
```

The ":DISPLAY:LABEL ON" in the above example is called a *program message*. Program messages are explained in more detail in "[Program Message Syntax](#)" on page 1335.

## Initializing the Interface and the Oscilloscope

To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. When using the Keysight VISA COM library, you can use the resource session object's Clear method to clear the interface buffer:

```

Dim myMgr As VisaComLib.ResourceManager
Dim myScope As VisaComLib.FormattedIO488

Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488

' Open the connection to the oscilloscope. Get the VISA Address from the
' Keysight Connection Expert (installed with Keysight IO Libraries Suite
').
Set myScope.IO = myMgr.Open("<VISA Address>")

' Clear the interface buffer and set the interface timeout to 10 seconds
.
myScope.IO.Clear
myScope.IO.Timeout = 10000

```

When you are using GPIB, CLEAR also resets the oscilloscope's parser. The parser is the program which reads in the instructions which you send it.

After clearing the interface, initialize the instrument to a preset state:

```
myScope.WriteString "*RST"
```

#### NOTE

#### Information for Initializing the Instrument

The actual commands and syntax for initializing the instrument are discussed in [Chapter 5](#), “Common (\*) Commands,” starting on page 177.

Refer to the Keysight IO Libraries Suite documentation for information on initializing the interface.

---

### Using :AUToscale to Automate Oscilloscope Setup

The :AUToscale command performs a very useful function for unknown waveforms by setting up the vertical channel, time base, and trigger level of the instrument.

The syntax for the autoscale command is:

```
myScope.WriteString ":AUToscale"
```

### Using Other Oscilloscope Setup Commands

A typical oscilloscope setup would set the vertical range and offset voltage, the horizontal range, delay time, delay reference, trigger mode, trigger level, and slope. An example of the commands that might be sent to the oscilloscope are:

```

myScope.WriteString ":CHANnel1:PROBe 10"
myScope.WriteString ":CHANnel1:RANGE 16"
myScope.WriteString ":CHANnel1:OFFSet 1.00"
myScope.WriteString ":TIMEbase:MODE MAIN"
myScope.WriteString ":TIMEbase:RANGE 1E-3"
myScope.WriteString ":TIMEbase:DELay 100E-6"

```

Vertical is set to 16 V full-scale (2 V/div) with center of screen at 1 V and probe attenuation set to 10. This example sets the time base at 1 ms full-scale (100 ms/div) with a delay of 100  $\mu$ s.

### Example Oscilloscope Setup Code

This program demonstrates the basic command structure used to program the oscilloscope.

```
' Initialize the instrument interface to a known state.
myScope.IO.Clear
myScope.IO.Timeout = 10000      ' Set interface timeout to 10 seconds.

' Initialize the instrument to a preset state.
myScope.WriteString "*RST"

' Set the time base mode to normal with the horizontal time at
' 50 ms/div with 0 s of delay referenced at the center of the
' graticule.
myScope.WriteString ":TIMEbase:RANGE 5E-4"      ' Time base to 50 us/div.
myScope.WriteString ":TIMEbase:DELay 0"           ' Delay to zero.
myScope.WriteString ":TIMEbase:REference CENTER"   ' Display ref. at
                                                ' center.

' Set the vertical range to 1.6 volts full scale with center screen
' at -0.4 volts with 10:1 probe attenuation and DC coupling.
myScope.WriteString ":CHANnel1:PROBe 10"          ' Probe attenuation
                                                ' to 10:1.
myScope.WriteString ":CHANnel1:RANGE 1.6"          ' Vertical range
                                                ' 1.6 V full scale.
myScope.WriteString ":CHANnel1:OFFSet -0.4"        ' Offset to -0.4.
myScope.WriteString ":CHANnel1:COUPling DC"        ' Coupling to DC.

' Configure the instrument to trigger at -0.4 volts with normal
' triggering.
myScope.WriteString ":TRIGger:SWEep NORMAL"       ' Normal triggering.
myScope.WriteString ":TRIGger:LEVel -0.4"          ' Trigger level to -0.4.
myScope.WriteString ":TRIGger:SLOPe POSitive"      ' Trigger on pos. slope.

' Configure the instrument for normal acquisition.
myScope.WriteString ":ACQuire:TYPE NORMAL"        ' Normal acquisition.
```

### Capturing Data with the :DIGITIZE Command

The :DIGITIZE command captures data that meets the specifications set up by the :ACQUIRE subsystem. When the digitize process is complete, the acquisition is stopped. The captured data can then be measured by the instrument or transferred to the controller for further analysis. The captured data consists of two parts: the waveform data record, and the preamble.

**NOTE****Ensure New Data is Collected**

When you change the oscilloscope configuration, the waveform buffers are cleared. Before doing a measurement, send the :DIGItize command to the oscilloscope to ensure new data has been collected.

When you send the :DIGItize command to the oscilloscope, the specified channel signal is digitized with the current :ACQuire parameters. To obtain waveform data, you must specify the :WAVeform parameters for the SOURce channel, the FORMat type, and the number of POINTs prior to sending the :WAVeform:DATA? query.

**NOTE****Set :TIMEbase:MODE to MAIN when using :DIGItize**

:TIMEbase:MODE must be set to MAIN to perform a :DIGItize command or to perform any :WAVeform subsystem query. A "Settings conflict" error message will be returned if these commands are executed when MODE is set to ROLL, XY, or WINDow (zoomed). Sending the \*RST (reset) command will also set the time base mode to normal.

The number of data points comprising a waveform varies according to the number requested in the :ACQuire subsystem. The :ACQuire subsystem determines the number of data points, type of acquisition, and number of averages used by the :DIGItize command. This allows you to specify exactly what the digitized information contains.

The following program example shows a typical setup:

```
myScope.WriteString ":ACQuire:TYPE AVERage"
myScope.WriteString ":ACQuire:COMPLETE 100"
myScope.WriteString ":ACQuire:COUNT 8"
myScope.WriteString ":DIGItize CHANnel1"
myScope.WriteString ":WAVeform:SOURce CHANnel1"
myScope.WriteString ":WAVeform:FORMAT BYTE"
myScope.WriteString ":WAVeform:POINTS 500"
myScope.WriteString ":WAVeform:DATA?"
```

This setup places the instrument into the averaged mode with eight averages. This means that when the :DIGItize command is received, the command will execute until the signal has been averaged at least eight times.

After receiving the :WAVeform:DATA? query, the instrument will start passing the waveform information.

Digitized waveforms are passed from the instrument to the controller by sending a numerical representation of each digitized point. The format of the numerical representation is controlled with the :WAVeform:FORMAT command and may be selected as BYTE, WORD, or ASCii.

The easiest method of transferring a digitized waveform depends on data structures, formatting available and I/O capabilities. You must scale the integers to determine the voltage value of each point. These integers are passed starting with the left most point on the instrument's display.

For more information, see the waveform subsystem commands and corresponding program code examples in [Chapter 34](#), “:WAVeform Commands,” starting on page 1145.

**NOTE****Aborting a Digitize Operation Over the Programming Interface**

When using the programming interface, you can abort a digitize operation by sending a Device Clear over the bus (for example, `myScope.IO.Clear`).

## Reading Query Responses from the Oscilloscope

After receiving a query (command header followed by a question mark), the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the interface to the designated listener (typically a controller).

The statement for reading a query response message from an instrument's output queue typically has a format specification for handling the response message.

When using the VISA COM library in Visual Basic, you use different read methods (`ReadString`, `ReadNumber`, `ReadList`, or `ReadIEEEBlock`) for the various query response formats. For example, to read the result of the query command `:CHANnel1:COUpling?` you would execute the statements:

```
myScope.WriteString ":CHANnel1:COUpling?"
Dim strQueryResult As String
strQueryResult = myScope.ReadString
```

This reads the current setting for the channel one coupling into the string variable `strQueryResult`.

All results for queries (sent in one program message) must be read before another program message is sent.

Sending another command before reading the result of the query clears the output buffer and the current response. This also causes an error to be placed in the error queue.

Executing a read statement before sending a query causes the controller to wait indefinitely.

The format specification for handling response messages depends on the programming language.

## Reading Query Results into String Variables

The output of the instrument may be numeric or character data depending on what is queried. Refer to the specific command descriptions for the formats and types of data returned from queries.

### NOTE

#### Express String Variables Using Exact Syntax

In Visual Basic, string variables are case sensitive and must be expressed exactly the same each time they are used.

The following example shows numeric data being returned to a string variable:

```
myScope.WriteString ":CHANnel1:RANGE?"
Dim strQueryResult As String
strQueryResult = myScope.ReadString
MsgBox "Range (string):" + strQueryResult
```

After running this program, the controller displays:

**Range (string): +40.0E+00**

## Reading Query Results into Numeric Variables

The following example shows numeric data being returned to a numeric variable:

```
myScope.WriteString ":CHANnel1:RANGE?"
Dim varQueryResult As Variant
varQueryResult = myScope.ReadNumber
MsgBox "Range (variant):" + CStr(varQueryResult)
```

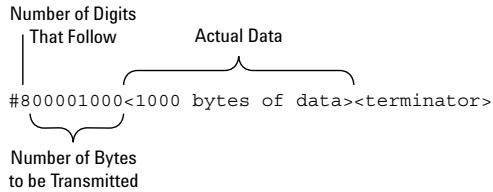
After running this program, the controller displays:

**Range (variant): 40**

## Reading Definite-Length Block Query Response Data

Definite-length block query response data allows any type of device-dependent data to be transmitted over the system interface as a series of 8-bit binary data bytes. This is particularly useful for sending large quantities of data or 8-bit extended ASCII codes. The syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the decimal integer. After the non-zero digit is the decimal integer that states the number of 8-bit data bytes being sent. This is followed by the actual data.

For example, for transmitting 1000 bytes of data, the syntax would be:



**Figure 2** Definite-length block response data

The "8" states the number of digits that follow, and "00001000" states the number of bytes to be transmitted.

The VISA COM library's ReadIEEEBlock and WriteIEEEBlock methods understand the definite-length block syntax, so you can simply use variables that contain the data:

```
' Read oscilloscope setup using ":SYSTem:SETup?" query.
myScope.WriteString ":SYSTem:SETup?"
Dim varQueryResult As Variant
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)

' Write learn string back to oscilloscope using ":SYSTem:SETup" command:
myScope.WriteIEEEBlock ":SYSTem:SETup ", varQueryResult
```

## Sending Multiple Queries and Reading Results

You can send multiple queries to the instrument within a single command string, but you must also read them back as a single query result. This can be accomplished by reading them back into a single string variable, multiple string variables, or multiple numeric variables.

For example, to read the :TIMEbase:RANGE?;DElay? query result into a single string variable, you could use the commands:

```
myScope.WriteString ":TIMEbase:RANGE?;DElay?"
Dim strQueryResult As String
strQueryResult = myScope.ReadString
MsgBox "Timebase range; delay:" + strQueryResult
```

When you read the result of multiple queries into a single string variable, each response is separated by a semicolon. For example, the output of the previous example would be:

```
Timebase range; delay: <range_value>;<delay_value>
```

To read the :TIMEbase:RANGE?;DElay? query result into multiple string variables, you could use the ReadList method to read the query results into a string array variable using the commands:

```
myScope.WriteString ":TIMEbase:RANGE?;DElay?"
Dim strResults() As String
```

```
strResults() = myScope.ReadList(ASCIIIType_BSTR)
MsgBox "Timebase range: " + strResults(0) + ", delay: " + strResults(1)
```

To read the :TIMEbase:RANGE?;DElay? query result into multiple numeric variables, you could use the ReadList method to read the query results into a variant array variable using the commands:

```
myScope.WriteString ":TIMEbase:RANGE?;DElay?"
Dim varResults() As Variant
varResults() = myScope.ReadList
MsgBox "Timebase range: " + FormatNumber(varResults(0) * 1000, 4) +
       " ms, delay: " + FormatNumber(varResults(1) * 1000000, 4) + " us"
```

## Checking Instrument Status

Status registers track the current status of the instrument. By checking the instrument status, you can find out whether an operation has been completed, whether the instrument is receiving triggers, and more.

For more information, see [Chapter 39](#), “Status Reporting,” starting on page 1299 which explains how to check the status of the instrument.

## Other Ways of Sending Commands

Standard Commands for Programmable Instrumentation (SCPI) can also be sent via a Telnet socket or through the Browser Web Control:

- "Telnet Sockets" on page 67
- "Sending SCPI Commands Using Browser Web Control" on page 67

### Telnet Sockets

The following information is provided for programmers who wish to control the oscilloscope with SCPI commands in a Telnet session.

To connect to the oscilloscope via a telnet socket, issue the following command:

```
telnet <hostname> 5024
```

where <hostname> is the hostname of the oscilloscope. This will give you a command line with prompt.

For a command line without a prompt, use port 5025. For example:

```
telnet <hostname> 5025
```

### Sending SCPI Commands Using Browser Web Control

To send SCPI commands using the Browser Web Control feature, establish a connection to the oscilloscope via LAN as described in the *InfiniiVision 3000T X-Series Oscilloscopes User's Guide*. When you make the connection to the oscilloscope via LAN and the instrument's welcome page is displayed, select the **Browser Web Control** tab, then select the **Remote Programming** link.



## 4 Commands Quick Reference

Command Summary / 70

Syntax Elements / 174

## Command Summary

- Common (\*) Commands Summary (see [page 72](#))
- Root (:) Commands Summary (see [page 75](#))
- :ACQuire Commands Summary (see [page 78](#))
- :BUS<n> Commands Summary (see [page 79](#))
- :CALibrate Commands Summary (see [page 80](#))
- :CHANnel<n> Commands Summary (see [page 81](#))
- :COUNter Commands Summary (see [page 83](#))
- :DEMO Commands Summary (see [page 84](#))
- :DIGItal<n> Commands Summary (see [page 84](#))
- :DISPlay Commands Summary (see [page 85](#))
- :DVM Commands Summary (see [page 86](#))
- :EXTernal Trigger Commands Summary (see [page 87](#))
- :FFT Commands Summary (see [page 87](#))
- :FUNCtion Commands Summary (see [page 89](#))
- :HARDcopy Commands Summary (see [page 92](#))
- :LISTer Commands Summary (see [page 94](#))
- :MARKer Commands Summary (see [page 94](#))
- :MEASure Commands Summary (see [page 96](#))
- :MEASure Power Commands Summary (see [page 111](#))
- :MTEST Commands Summary (see [page 115](#))
- :POD<n> Commands Summary (see [page 117](#))
- :POWER Commands Summary (see [page 118](#))
- :RECall Commands Summary (see [page 124](#))
- :SAVE Commands Summary (see [page 125](#))
- General :SBUS<n> Commands Summary (see [page 128](#))
- :SBUS<n>:A429 Commands Summary (see [page 128](#))
- :SBUS<n>:CAN Commands Summary (see [page 130](#))
- :SBUS<n>:FLEXray Commands Summary (see [page 132](#))
- :SBUS<n>:I2S Commands Summary (see [page 133](#))
- :SBUS<n>:IIC Commands Summary (see [page 135](#))
- :SBUS<n>:LIN Commands Summary (see [page 136](#))
- :SBUS<n>:M1553 Commands Summary (see [page 138](#))
- :SBUS<n>:SENT Commands Summary (see [page 139](#))

- :SBUS<n>:SPI Commands Summary (see [page 141](#))
- :SBUS<n>:UART Commands Summary (see [page 142](#))
- General :SEARch Commands Summary (see [page 144](#))
- :SEARch:EDGE Commands Summary (see [page 145](#))
- :SEARch:GLITch Commands Summary (see [page 145](#))
- :SEARch:PEAK Commands Summary (see [page 146](#))
- :SEARch:RUNT Commands Summary (see [page 146](#))
- :SEARch:TRANSition Commands Summary (see [page 146](#))
- :SEARch:SERial:A429 Commands Summary (see [page 147](#))
- :SEARch:SERial:CAN Commands Summary (see [page 148](#))
- :SEARch:SERial:FLEXray Commands Summary (see [page 148](#))
- :SEARch:SERial:I2S Commands Summary (see [page 149](#))
- :SEARch:SERial:IIC Commands Summary (see [page 150](#))
- :SEARch:SERial:LIN Commands Summary (see [page 150](#))
- :SEARch:SERial:M1553 Commands Summary (see [page 151](#))
- :SEARch:SERial:SENT Commands Summary (see [page 152](#))
- :SEARch:SERial:SPI Commands Summary (see [page 152](#))
- :SEARch:SERial:UART Commands Summary (see [page 153](#))
- :SYSTem Commands Summary (see [page 153](#))
- :TIMEbase Commands Summary (see [page 155](#))
- General :TRIGger Commands Summary (see [page 156](#))
- :TRIGger:DELay Commands Summary (see [page 157](#))
- :TRIGger:EBURst Commands Summary (see [page 157](#))
- :TRIGger[:EDGE] Commands Summary (see [page 158](#))
- :TRIGger:GLITch Commands Summary (see [page 159](#))
- :TRIGger:NFC Commands Summary (see [page 160](#))
- :TRIGger:OR Commands Summary (see [page 161](#))
- :TRIGger:PATTERn Commands Summary (see [page 161](#))
- :TRIGger:RUNT Commands Summary (see [page 162](#))
- :TRIGger:SHOLd Commands Summary (see [page 163](#))
- :TRIGger:TRANSition Commands Summary (see [page 163](#))
- :TRIGger:TV Commands Summary (see [page 164](#))
- :TRIGger:USB Commands Summary (see [page 165](#))
- :TRIGger:ZONE Commands Summary (see [page 165](#))
- :WAVeform Commands Summary (see [page 166](#))

- :WGEN Commands Summary (see [page 169](#))
- :WMEMory<r> Commands Summary (see [page 172](#))

**Table 2** Common (\*) Commands Summary

Command	Query	Options and Query Returns
*CLS (see <a href="#">page 181</a> )	n/a	n/a
*ESE <mask> (see <a href="#">page 182</a> )	*ESE? (see <a href="#">page 183</a> )	<mask> ::= 0 to 255; an integer in NR1 format:  Bit Weight Name Enables ---- ----- ---- 7 128 PON Power On 6 64 URQ User Request 5 32 CME Command Error 4 16 EXE Execution Error 3 8 DDE Dev. Dependent Error 2 4 QYE Query Error 1 2 RQL Request Control 0 1 OPC Operation Complete
n/a	*ESR? (see <a href="#">page 184</a> )	<status> ::= 0 to 255; an integer in NR1 format
n/a	*IDN? (see <a href="#">page 184</a> )	KEYSIGHT TECHNOLOGIES, <model>, <serial number>, X.XX.XX <model> ::= the model number of the instrument <serial number> ::= the serial number of the instrument <X.XX.XX> ::= the software revision of the instrument
n/a	*LRN? (see <a href="#">page 187</a> )	<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format
*OPC (see <a href="#">page 188</a> )	*OPC? (see <a href="#">page 188</a> )	ASCII "1" is placed in the output queue when all pending device operations have completed.

**Table 2** Common (\*) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	*OPT? (see <a href="#">page 189</a> )	<pre> &lt;return_value&gt; ::= 0,0,&lt;license info&gt; &lt;license info&gt; ::= &lt;All field&gt;, &lt;reserved&gt;, &lt;MSO&gt;, &lt;Xilinx FPGA Probe&gt;, &lt;Memory&gt;, &lt;Low Speed Serial&gt;, &lt;Automotive Serial&gt;, &lt;reserved&gt;, &lt;FlexRay Serial&gt;, &lt;Power Measurements&gt;, &lt;RS-232/UART Serial&gt;, &lt;reserved&gt;, &lt;Mask Test&gt;, &lt;reserved&gt;, &lt;Bandwidth&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;I2S Serial&gt;, &lt;reserved&gt;, &lt;Educator's Kit&gt;, &lt;Waveform Generator&gt;, &lt;MIL-1553/ARINC 429 Serial&gt;, &lt;Extended Video&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;Digital Voltmeter&gt;, &lt;Spectrum Visualizer&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;USB 2.0 Low/Full Speed&gt;, &lt;USB 2.0 High Speed&gt; &lt;All field&gt; ::= {0   All} &lt;reserved&gt; ::= 0 &lt;MSO&gt; ::= {0   MSO} &lt;Xilinx FPGA Probe&gt; ::= {0   FPGAX} &lt;Memory&gt; ::= {0   MEMUP} &lt;Low Speed Serial&gt; ::= {0   EMBD} &lt;Automotive Serial&gt; ::= {0   AUTO} &lt;FlexRay Serial&gt; ::= {0   FLEX} &lt;Power Measurements&gt; ::= {0   PWR} &lt;RS-232/UART Serial&gt; ::= {0   COMP} &lt;Mask Test&gt; ::= {0   MASK} &lt;Bandwidth&gt; ::= {0   BW20   BW50} &lt;I2S Serial&gt; ::= {0   AUDIO} &lt;Educator's Kit&gt; ::= {0   EDK} &lt;Waveform Generator&gt; ::= {0   WAVEGEN} &lt;MIL-1553/ARINC 429 Serial&gt; ::= {0   AERO} &lt;Extended Video&gt; ::= {0   VID} &lt;Digital Voltmeter&gt; ::= {0   DVM} &lt;Spectrum Visualizer&gt; ::= {0   ASV} &lt;USB 2.0 Low/Full Speed&gt; ::= {0   USF} &lt;USB 2.0 High Speed&gt; ::= {0   HSF} </pre>

**Table 2** Common (\*) Commands Summary (continued)

Command	Query	Options and Query Returns																																				
*RCL <value> (see page 191)	n/a	<value> ::= { 0   1   4   5   6   7   8   9 }																																				
*RST (see page 192)	n/a	See *RST (Reset) (see page 192)																																				
*SAV <value> (see page 195)	n/a	<value> ::= { 0   1   4   5   6   7   8   9 }																																				
*SRE <mask> (see page 196)	*SRE? (see page 197)	<p>&lt;mask&gt; ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. &lt;mask&gt; ::= following values:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>Enables</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER</td> <td>Operation Status Reg</td> </tr> <tr> <td>6</td> <td>64</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB</td> <td>Event Status Bit</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV</td> <td>Message Available</td> </tr> <tr> <td>3</td> <td>8</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG</td> <td>Message</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR</td> <td>User</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG</td> <td>Trigger</td> </tr> </tbody> </table>	Bit	Weight	Name	Enables	7	128	OPER	Operation Status Reg	6	64	---	(Not used.)	5	32	ESB	Event Status Bit	4	16	MAV	Message Available	3	8	---	(Not used.)	2	4	MSG	Message	1	2	USR	User	0	1	TRG	Trigger
Bit	Weight	Name	Enables																																			
7	128	OPER	Operation Status Reg																																			
6	64	---	(Not used.)																																			
5	32	ESB	Event Status Bit																																			
4	16	MAV	Message Available																																			
3	8	---	(Not used.)																																			
2	4	MSG	Message																																			
1	2	USR	User																																			
0	1	TRG	Trigger																																			
n/a	*STB? (see page 198)	<p>&lt;value&gt; ::= 0 to 255; an integer in NR1 format, as shown in the following:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>"1" Indicates</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER</td> <td>Operation status condition occurred.</td> </tr> <tr> <td>6</td> <td>64</td> <td>RQS/</td> <td>Instrument is MSS requesting service.</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB</td> <td>Enabled event status condition occurred.</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV</td> <td>Message available.</td> </tr> <tr> <td>3</td> <td>8</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG</td> <td>Message displayed.</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR</td> <td>User event condition occurred.</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG</td> <td>A trigger occurred.</td> </tr> </tbody> </table>	Bit	Weight	Name	"1" Indicates	7	128	OPER	Operation status condition occurred.	6	64	RQS/	Instrument is MSS requesting service.	5	32	ESB	Enabled event status condition occurred.	4	16	MAV	Message available.	3	8	---	(Not used.)	2	4	MSG	Message displayed.	1	2	USR	User event condition occurred.	0	1	TRG	A trigger occurred.
Bit	Weight	Name	"1" Indicates																																			
7	128	OPER	Operation status condition occurred.																																			
6	64	RQS/	Instrument is MSS requesting service.																																			
5	32	ESB	Enabled event status condition occurred.																																			
4	16	MAV	Message available.																																			
3	8	---	(Not used.)																																			
2	4	MSG	Message displayed.																																			
1	2	USR	User event condition occurred.																																			
0	1	TRG	A trigger occurred.																																			
*TRG (see page 200)	n/a	n/a																																				
n/a	*TST? (see page 201)	<result> ::= 0 or non-zero value; an integer in NR1 format																																				
*WAI (see page 202)	n/a	n/a																																				

**Table 3** Root (:) Commands Summary

Command	Query	Options and Query Returns
:ACTivity (see page 207)	:ACTivity? (see page 207)	<return value> ::= <edges>,<levels> <edges> ::= presence of edges (32-bit integer in NR1 format) <levels> ::= logical highs or lows (32-bit integer in NR1 format)
n/a	:AER? (see page 208)	{0   1}; an integer in NR1 format
:AUToscale [<source>[,...<source>]] (see page 209)	n/a	<source> ::= CHANnel<n> for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   POD1   POD2} for MSO models <source> can be repeated up to 5 times <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:AUToscale:AMODE <value> (see page 211)	:AUToscale:AMODE? (see page 211)	<value> ::= {NORMAl   CURRent}
:AUToscale:CHANnels <value> (see page 212)	:AUToscale:CHANnels? (see page 212)	<value> ::= {ALL   DISPlayed}
:AUToscale:FDEBug {{0   OFF}   {1   ON}} (see page 213)	:AUToscale:FDEBug? (see page 213)	{0   1}
:BLANK [<source>] (see page 214)	n/a	<source> ::= {CHANnel<n>}   FUNCtion<m>   MATH<m>   FFT   SBUS{1   2}   WMEMory<r>} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   POD{1   2}   BUS{1   2}   FUNCtion<m>   MATH<m>   FFT   SBUS{1   2}   WMEMory<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format

**Table 3** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGItize [<source>[, . . . , <source>]] (see <a href="#">page 215</a> )	n/a	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}} for MSO models</p> <p>&lt;source&gt; can be repeated up to 5 times</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:HWEenable <n> (see <a href="#">page 217</a> )	:HWEenable? (see <a href="#">page 217</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:HWERregister:CONDiti on? (see <a href="#">page 219</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:HWERregister[:EVENT]? (see <a href="#">page 220</a> )	<n> ::= 16-bit integer in NR1 format
:MTEenable <n> (see <a href="#">page 221</a> )	:MTEenable? (see <a href="#">page 221</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:MTERegister[:EVENT]? (see <a href="#">page 223</a> )	<n> ::= 16-bit integer in NR1 format
:OPEE <n> (see <a href="#">page 225</a> )	:OPEE? (see <a href="#">page 226</a> )	<n> ::= 15-bit integer in NR1 format
n/a	:OPERregister:CONDiti on? (see <a href="#">page 227</a> )	<n> ::= 15-bit integer in NR1 format
n/a	:OPERregister[:EVENT]? (see <a href="#">page 229</a> )	<n> ::= 15-bit integer in NR1 format

**Table 3** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns																																	
:OVLenable <mask> (see <a href="#">page 231</a> )	:OVLenable? (see <a href="#">page 232</a> )	<p>&lt;mask&gt; ::= 16-bit integer in NR1 format as shown:</p> <table> <thead> <tr> <th>Bit</th><th>Weight</th><th>Input</th></tr> </thead> <tbody> <tr><td>10</td><td>1024</td><td>Ext Trigger Fault</td></tr> <tr><td>9</td><td>512</td><td>Channel 4 Fault</td></tr> <tr><td>8</td><td>256</td><td>Channel 3 Fault</td></tr> <tr><td>7</td><td>128</td><td>Channel 2 Fault</td></tr> <tr><td>6</td><td>64</td><td>Channel 1 Fault</td></tr> <tr><td>4</td><td>16</td><td>Ext Trigger OVL</td></tr> <tr><td>3</td><td>8</td><td>Channel 4 OVL</td></tr> <tr><td>2</td><td>4</td><td>Channel 3 OVL</td></tr> <tr><td>1</td><td>2</td><td>Channel 2 OVL</td></tr> <tr><td>0</td><td>1</td><td>Channel 1 OVL</td></tr> </tbody> </table>	Bit	Weight	Input	10	1024	Ext Trigger Fault	9	512	Channel 4 Fault	8	256	Channel 3 Fault	7	128	Channel 2 Fault	6	64	Channel 1 Fault	4	16	Ext Trigger OVL	3	8	Channel 4 OVL	2	4	Channel 3 OVL	1	2	Channel 2 OVL	0	1	Channel 1 OVL
Bit	Weight	Input																																	
10	1024	Ext Trigger Fault																																	
9	512	Channel 4 Fault																																	
8	256	Channel 3 Fault																																	
7	128	Channel 2 Fault																																	
6	64	Channel 1 Fault																																	
4	16	Ext Trigger OVL																																	
3	8	Channel 4 OVL																																	
2	4	Channel 3 OVL																																	
1	2	Channel 2 OVL																																	
0	1	Channel 1 OVL																																	
n/a	:OVLRegister? (see <a href="#">page 233</a> )	<value> ::= integer in NR1 format. See OVLenable for <value>																																	
:PRINT [<options>] (see <a href="#">page 235</a> )	n/a	<p>&lt;options&gt; ::= [&lt;print option&gt;] [,...,&lt;print option&gt;]</p> <p>&lt;print option&gt; ::= {COLor   GRAYscale   PRINTER0   BMP8bit   BMP   PNG   NOFactors   FACTors}</p> <p>&lt;print option&gt; can be repeated up to 5 times.</p>																																	
:RUN (see <a href="#">page 236</a> )	n/a	n/a																																	
n/a	:SERial (see <a href="#">page 237</a> )	<return value> ::= unquoted string containing serial number																																	
:SINGle (see <a href="#">page 238</a> )	n/a	n/a																																	
n/a	:STATus? <display> (see <a href="#">page 239</a> )	<p>{0   1}</p> <p>&lt;display&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p>																																	
:STOP (see <a href="#">page 240</a> )	n/a	n/a																																	

**Table 3** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:TER? (see <a href="#">page 241</a> )	{0   1}
:VIEW <source> (see <a href="#">page 242</a> )	n/a	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}   WMEMORY&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}   WMEMORY&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p>

**Table 4** :ACQuire Commands Summary

Command	Query	Options and Query Returns
:ACQuire:COMplete <complete> (see <a href="#">page 245</a> )	:ACQuire:COMplete? (see <a href="#">page 245</a> )	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNT <count> (see <a href="#">page 246</a> )	:ACQuire:COUNT? (see <a href="#">page 246</a> )	<count> ::= an integer from 2 to 65536 in NR1 format
:ACQuire:MODE <mode> (see <a href="#">page 247</a> )	:ACQuire:MODE? (see <a href="#">page 247</a> )	<mode> ::= {RTIMe   ETIMe   SEGmented}
n/a	:ACQuire:POINTS? (see <a href="#">page 248</a> )	<# points> ::= an integer in NR1 format
:ACQuire:SEGmented:AN ALyze (see <a href="#">page 249</a> )	n/a	n/a (with Option SGM)
:ACQuire:SEGmented:CO UNT <count> (see <a href="#">page 250</a> )	:ACQuire:SEGmented:CO UNT? (see <a href="#">page 250</a> )	<count> ::= an integer from 2 to 1000 in NR1 format (with Option SGM)
:ACQuire:SEGmented:IN Dex <index> (see <a href="#">page 251</a> )	:ACQuire:SEGmented:IN Dex? (see <a href="#">page 251</a> )	<index> ::= an integer from 1 to 1000 in NR1 format (with Option SGM)

**Table 4** :ACQuire Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:ACQuire:SRATE? (see page 254)	<sample_rate> ::= sample rate (samples/s) in NR3 format
:ACQuire:TYPE <type> (see page 255)	:ACQuire:TYPE? (see page 255)	<type> ::= {NORMal   AVERage   HRESolution   PEAK}

**Table 5** :BUS<n> Commands Summary

Command	Query	Options and Query Returns
:BUS<n>:BIT<m> {{0   OFF}   {1   ON}} (see page 259)	:BUS<n>:BIT<m>? (see page 259)	{0   1} <n> ::= 1 or 2; an integer in NR1 format <m> ::= 0-15; an integer in NR1 format
:BUS<n>:BITS <channel_list>, {{0   OFF}   {1   ON}} (see page 260)	:BUS<n>:BITS? (see page 260)	<channel_list>, {0   1} <channel_list> ::= (@<m>,<m>:<m> ...) where "," is separator and ":" is range <n> ::= 1 or 2; an integer in NR1 format <m> ::= 0-15; an integer in NR1 format
:BUS<n>:CLEar (see page 262)	n/a	<n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:DISPlay {{0   OFF}   {1   ON}} (see page 263)	:BUS<n>:DISPlay? (see page 263)	{0   1} <n> ::= 1 or 2; an integer in NR1 format

**Table 5** :BUS<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:BUS<n>:LABEL <string> (see <a href="#">page 264</a> )	:BUS<n>:LABEL? (see <a href="#">page 264</a> )	<string> ::= quoted ASCII string up to 10 characters <n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:MASK <mask> (see <a href="#">page 265</a> )	:BUS<n>:MASK? (see <a href="#">page 265</a> )	<mask> ::= 32-bit integer in decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal <n> ::= 1 or 2; an integer in NR1 format

**Table 6** :CALibrate Commands Summary

Command	Query	Options and Query Returns
n/a	:CALibrate:DATE? (see <a href="#">page 269</a> )	<return value> ::= <year>,<month>,<day>; all in NR1 format
:CALibrate:LABEL <string> (see <a href="#">page 270</a> )	:CALibrate:LABEL? (see <a href="#">page 270</a> )	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:OUTPut <signal> (see <a href="#">page 271</a> )	:CALibrate:OUTPut? (see <a href="#">page 271</a> )	<signal> ::= {TRIGgers   MASK   WAVEgen   WGEN1   WGEN2} Note: WAVE and WGEN1 are equivalent. Note: WGEN2 only available on models with 2 WaveGen outputs.
n/a	:CALibrate:PROTected? (see <a href="#">page 273</a> )	{PROTected   UNPROtected}
:CALibrate:START (see <a href="#">page 274</a> )	n/a	n/a

**Table 6** :CALibrate Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:CALibrate:STATUS? (see <a href="#">page 275</a> )	<return value> ::= <status_code>, <status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:TEMPERature? (see <a href="#">page 276</a> )	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see <a href="#">page 277</a> )	<return value> ::= <hours>, <minutes>, <seconds>; all in NR1 format

**Table 7** :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BWLimit { {0   OFF}   {1   ON} } (see <a href="#">page 282</a> )	:CHANnel<n>:BWLimit? (see <a href="#">page 282</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:COUPling <coupling> (see <a href="#">page 283</a> )	:CHANnel<n>:COUPling? (see <a href="#">page 283</a> )	<coupling> ::= {AC   DC} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 284</a> )	:CHANnel<n>:DISPlay? (see <a href="#">page 284</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:IMPedance <impedance> (see <a href="#">page 285</a> )	:CHANnel<n>:IMPedance? (see <a href="#">page 285</a> )	<impedance> ::= {ONEMeg   FIFTy} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:INVert { {0   OFF}   {1   ON} } (see <a href="#">page 286</a> )	:CHANnel<n>:INVert? (see <a href="#">page 286</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:LABel <string> (see <a href="#">page 287</a> )	:CHANnel<n>:LABEL? (see <a href="#">page 287</a> )	<string> ::= any series of 32 or less ASCII characters enclosed in quotation marks <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:OFFSet <offset>[suffix] (see <a href="#">page 288</a> )	:CHANnel<n>:OFFSet? (see <a href="#">page 288</a> )	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V   mV} <n> ::= 1-2 or 1-4; in NR1 format

**Table 7** :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe <attenuation> (see <a href="#">page 289</a> )	:CHANnel<n>:PROBe? (see <a href="#">page 289</a> )	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4r in NR1 format
:CHANnel<n>:PROBe:HEA D [:TYPE] <head_param> (see <a href="#">page 290</a> )	:CHANnel<n>:PROBe:HEA D [:TYPE]? (see <a href="#">page 290</a> )	<head_param> ::= {SEND0   SEND6   SEND12   SEND20   DIFF0   DIFF6   DIFF12   DIFF20   NONE} <n> ::= 1 to (# analog channels) in NR1 format
n/a	:CHANnel<n>:PROBe:ID? (see <a href="#">page 291</a> )	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:MMO Del <value> (see <a href="#">page 292</a> )	:CHANnel<n>:PROBe:MMO Del? (see <a href="#">page 292</a> )	<value> ::= {P5205   P5210   P6205   P6241   P6243   P6245   P6246   P6247   P6248   P6249   P6250   P6251   P670X   P671X   TCP202} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:SKE W <skew_value> (see <a href="#">page 293</a> )	:CHANnel<n>:PROBe:SKE W? (see <a href="#">page 293</a> )	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:STY Pe <signal type> (see <a href="#">page 294</a> )	:CHANnel<n>:PROBe:STY Pe? (see <a href="#">page 294</a> )	<signal type> ::= {DIFFerential   SINGle} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROTection (see <a href="#">page 295</a> )	:CHANnel<n>:PROTection? (see <a href="#">page 295</a> )	{NORM   TRIP} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:RANGE <range>[suffix] (see <a href="#">page 296</a> )	:CHANnel<n>:RANGE? (see <a href="#">page 296</a> )	<range> ::= Vertical full-scale range value in NR3 format [suffix] ::= {V   mV} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:SCALE <scale>[suffix] (see <a href="#">page 297</a> )	:CHANnel<n>:SCALE? (see <a href="#">page 297</a> )	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V   mV} <n> ::= 1 to (# analog channels) in NR1 format

**Table 7** :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:UNITS <units> (see page 298)	:CHANnel<n>:UNITS? (see page 298)	<units> ::= {VOLT   AMPere} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:VERNier { {0   OFF}   {1   ON} } (see page 299)	:CHANnel<n>:VERNier? (see page 299)	{0   1} <n> ::= 1 to (# analog channels) in NR1 format

**Table 8** :COUNter Commands Summary

Command	Query	Options and Query Returns
n/a	:COUNTER:CURREnt? (see page 303)	<value> ::= current counter value in NR3 format
:COUNTER:ENABLE { {0   OFF}   {1   ON} } (see page 304)	:COUNTER:ENABLE? (see page 304)	{0   1}
:COUNTER:MODE <mode> (see page 305)	:COUNTER:MODE (see page 305)	<mode> ::= {FREQuency   PERiod   TOTalize}
:COUNTER:NDIGits <value> (see page 306)	:COUNTER:NDIGits (see page 306)	<value> ::= 3 to 8 in NR1 format
:COUNTER:SOURce <source> (see page 307)	:COUNTER:SOURce? (see page 307)	<source> ::= {CHANnel<n>   TQEEvent} <n> ::= 1 to (# analog channels) in NR1 format
:COUNTER:TOTalize:CLEar (see page 308)	n/a	n/a
:COUNTER:TOTalize:GATE:ENABLE { {0   OFF}   {1   ON} } (see page 309)	:COUNTER:TOTalize:GATE:ENABLE? (see page 309)	{0   1}
:COUNTER:TOTalize:GATE:POLarity <polarity> (see page 310)	:COUNTER:TOTalize:GATE:POLarity? (see page 310)	<polarity> ::= {{NEGative   FALLing}   {POSitive   RISing}}
:COUNTER:TOTalize:GATE:SOURce <source> (see page 311)	:COUNTER:TOTalize:GATE:SOURce? (see page 311)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:COUNTER:TOTalize:SLOPe <slope> (see page 312)	:COUNTER:TOTalize:SLOPe? (see page 312)	<slope> ::= {{NEGative   FALLing}   {POSitive   RISing}}

**Table 9** :DEMO Commands Summary

Command	Query	Options and Query Returns
:DEMO:FUNCTION <signal> (see page 314)	:DEMO:FUNCTION? (see <a href="#">page 317</a> )	<signal> ::= {SINusoid   NOISy   PHASe   RINGing   SINGle   AM   CLK   GLITch   BURSt   MSO   RUNT   TRANSition   RFBURst   SHOLD   LFSine   FMBurst   ETE   CAN   LIN   UART   I2C   SPI   I2S   CANLin   ARINC   FLEXray   MIL   MIL2   USB   NMONotonic}
:DEMO:FUNCTION:PHASE: PHASE <angle> (see page 318)	:DEMO:FUNCTION:PHASE: PHASE? (see <a href="#">page 318</a> )	<angle> ::= angle in degrees from 0 to 360 in NR3 format
:DEMO:OUTPut {{0   OFF}   {1   ON}} (see page 319)	:DEMO:OUTPut? (see <a href="#">page 319</a> )	{0   1}

**Table 10** :DIGItal<d> Commands Summary

Command	Query	Options and Query Returns
:DIGItal<d>:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 323</a> )	:DIGItal<d>:DISPlay? (see <a href="#">page 323</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format {0   1}
:DIGItal<d>:LABel <string> (see page 324)	:DIGItal<d>:LABel? (see <a href="#">page 324</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <string> ::= any series of 10 or less ASCII characters enclosed in quotation marks
:DIGItal<d>:POSIon <position> (see page 325)	:DIGItal<d>:POSITION? (see <a href="#">page 325</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <position> ::= 0-7 if display size = large, 0-15 if size = medium, 0-31 if size = small Returns -1 when there is no space to display the digital waveform.

**Table 10** :DIGItal<d> Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGItal<d>:SIZE <value> (see <a href="#">page 326</a> )	:DIGItal<d>:SIZE? (see <a href="#">page 326</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {SMALL   MEDIUM   LARGE}
:DIGItal<d>:THreshold <value>[suffix] (see <a href="#">page 327</a> )	:DIGItal<d>:THreshold? ? (see <a href="#">page 327</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {CMOS   ECL   TTL   <user defined value>} <user defined value> ::= value in NR3 format from -8.00 to +8.00 [suffix] ::= {V   mV   uV}

**Table 11** :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n> {{0   OFF}   {1   ON}} (see <a href="#">page 332</a> )	:DISPlay:ANNotation<n>? (see <a href="#">page 332</a> )	{0   1} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:BACKground <mode> (see <a href="#">page 333</a> )	:DISPlay:ANNotation<n>:BACKground? (see <a href="#">page 333</a> )	<mode> ::= {OPAQUE   INVERTED   TRANSPARENT} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:COLOR <color> (see <a href="#">page 334</a> )	:DISPlay:ANNotation<n>:COLOR? (see <a href="#">page 334</a> )	<color> ::= {CH1   CH2   CH3   CH4   DIG   MATH   REF   MARKER   WHITE   RED} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:TEXT <string> (see <a href="#">page 335</a> )	:DISPlay:ANNotation<n>:TEXT? (see <a href="#">page 335</a> )	<string> ::= quoted ASCII string (up to 254 characters) <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:X1Position <value> (see <a href="#">page 336</a> )	:DISPlay:ANNotation<n>:X1Position? (see <a href="#">page 336</a> )	<value> ::= an integer from 0 to (800 - width of annotation) in NR1 format. <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:Y1Position <value> (see <a href="#">page 337</a> )	:DISPlay:ANNotation<n>:Y1Position? (see <a href="#">page 337</a> )	<value> ::= an integer from 0 to (480 - height of annotation) in NR1 format. <n> ::= an integer from 1 to 4 in NR1 format.

**Table 11** :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:CLEar (see page 338)	n/a	n/a
n/a	:DISPlay:DATA? [ <format>] [,] [&lt;palett e&gt;] (see page 339)</format>	<format> ::= {BMP   BMP8bit   PNG} <palette> ::= {COLOR   GRAYscale} <display data> ::= data in IEEE 488.2 # format
:DISPlay:INTensity:WA Veform <value> (see page 340)	:DISPlay:INTensity:WA Veform? (see page 340)	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:LABEL {{0   OFF}   {1   ON}} (see page 341)	:DISPlay:LABEL? (see page 341)	{0   1}
:DISPlay:LABList <binary block> (see page 342)	:DISPlay:LABList? (see page 342)	<binary block> ::= an ordered list of up to 75 labels, each 10 characters maximum, separated by newline characters
:DISPlay:MENU <menu> (see page 343)	n/a	<menu> ::= {MASK   MEASure   SEGmented   LISTer   POWER}
:DISPlay:SIDebar <sidebar> (see page 344)	n/a	<sidebar> ::= {SUMMARY   CURSors   MEASurements   DVM   NAVigate   CONTrols   EVENTS   COUNTER}
:DISPlay:PERSistence <value> (see page 345)	:DISPlay:PERSistence? (see page 345)	<value> ::= {MINimum   INFinite   <time>} <time> ::= seconds in in NR3 format from 100E-3 to 60E0
:DISPlay:VECTors {1   ON} (see page 346)	:DISPlay:VECTors? (see page 346)	1

**Table 12** :DVM Commands Summary

Command	Query	Options and Query Returns
:DVM:ARAnge {{0   OFF}   {1   ON}} (see page 348)	:DVM:ARAnge? (see page 348)	{0   1}
n/a	:DVM:CURREnt? (see page 349)	<dvm_value> ::= floating-point number in NR3 format
:DVM:ENABLE {{0   OFF}   {1   ON}} (see page 350)	:DVM:ENABLE? (see page 350)	{0   1}

**Table 12** :DVM Commands Summary (continued)

Command	Query	Options and Query Returns
:DVM:MODE <mode> (see page 351)	:DVM:MODE? (see page 351)	<dvm_mode> ::= {ACRMs   DC   DCRMs}
:DVM:SOURce <source> (see page 352)	:DVM:SOURce? (see page 352)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format

**Table 13** :EXTernal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXTernal:BWLImIt <bwlimit> (see page 354)	:EXTernal:BWLImIt? (see page 354)	<bwlimit> ::= {0   OFF}
:EXTernal:PROBe <attenuation> (see page 355)	:EXTernal:PROBe? (see page 355)	<attenuation> ::= probe attenuation ratio in NR3 format
:EXTernal:RANGE <range>[<suffix>] (see page 356)	:EXTernal:RANGE? (see page 356)	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V   mV}
:EXTernal:UNITS <units> (see page 357)	:EXTernal:UNITS? (see page 357)	<units> ::= {VOLT   AMPere}

**Table 14** :FFT Commands Summary

Command	Query	Options and Query Returns
:FFT:AVERage:COUNT <count> (see page 361)	:FFT:AVERage:COUNT? (see page 361)	<count> ::= an integer from 2 to 65536 in NR1 format.
:FFT:CENTer <frequency> (see page 362)	:FFT:CENTer? (see page 362)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.
:FFT:CLEar (see page 363)	n/a	n/a
:FFT:DISPlay {{0   OFF}   {1   ON}} (see page 364)	:FFT:DISPlay? (see page 364)	<s> ::= 1-6, in NR1 format. {0   1}
:FFT:DMODe <display_mode> (see page 365)	:FFT:DMODe? (see page 365)	<display_mode> ::= {NORMAL   AVERage   MAXHold   MINHold}

**Table 14** :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFT:FREQuency:STARt <frequency> (see page 367)	:FFT:FREQuency:STARt? (see page 367)	<frequency> ::= the start frequency in NR3 format.
:FFT:FREQuency:STOP <frequency> (see page 368)	:FFT:FREQuency:STOP? (see page 368)	<frequency> ::= the stop frequency in NR3 format.
:FFT:GATE <gating> (see page 369)	:FFT:GATE? (see page 369)	<gating> ::= {NONE   ZOOM}
:FFT:OFFSet <offset> (see page 370)	:FFT:OFFSet? (see page 370)	<offset> ::= the value at center screen in NR3 format.
:FFT:RANGe <range> (see page 371)	:FFT:RANGe? (see page 371)	<range> ::= the full-scale vertical axis value in NR3 format.
:FFT:REFerence <level> (see page 372)	:FFT:REFerence? (see page 372)	<level> ::= the current reference level in NR3 format.
:FFT:SCALe <scale value>[<suffix>] (see page 373)	:FFT:SCALe? (see page 373)	<scale_value> ::= integer in NR1 format. <suffix> ::= dB
:FFT:SOURce1 <source> (see page 374)	:FFT:SOURce1? (see page 374)	<source> ::= {CHANnel<n>   FUNCTION<c>   MATH<c>} <n> ::= 1 to (# analog channels) in NR1 format. <c> ::= {1   2}
:FFT:SPAN <span> (see page 375)	:FFT:SPAN? (see page 375)	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.
:FFT:VTYPe <units> (see page 376)	:FFT:VTYPe? (see page 376)	<units> ::= {DECibel   VRMS}
:FFT:WINDOW <>window> (see page 377)	:FFT:WINDOW? (see page 377)	<window> ::= {RECTangular   HANNing   FLATtop   BHARris}

**Table 15** :FUNCTION<m> Commands Summary

Command	Query	Options and Query Returns
:FUNCTION<m>:AVERage:COUNt <count> (see page 384)	:FUNCTION<m>:AVERage:COUNt? (see page 384)	<count> ::= an integer from 2 to 65536 in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:CLOCk <source> (see page 385)	:FUNCTION<m>:BUS:CLOCK? (see page 385)	<source> ::= {CHANnel<n>   DIGItal<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:SLOPe <slope> (see page 386)	:FUNCTION<m>:BUS:SLOPe? (see page 386)	<slope> ::= {NEGative   POSitive   EITHer} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YINCrement <value> (see page 387)	:FUNCTION<m>:BUS:YINCREMENT? (see page 387)	<value> ::= value per bus code, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YORigin <value> (see page 388)	:FUNCTION<m>:BUS:YORIGIN? (see page 388)	<value> ::= value at bus code = 0, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YUNits <units> (see page 389)	:FUNCTION<m>:BUS:YUNITS? (see page 389)	<units> ::= {VOLT   AMPere   NONE} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:CLEar (see page 390)	n/a	n/a
:FUNCTION<m>:DISPlay {{0   OFF}   {1   ON}} (see page 391)	:FUNCTION<m>:DISPLAY? (see page 391)	{0   1} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:CENTer <frequency> (see page 392)	:FUNCTION<m>[:FFT]:CENTER? (see page 392)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz. <m> ::= 1 to (# math functions) in NR1 format

**Table 15** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>[:FFT]:FREQUENCY:START <frequency> (see page 393)	:FUNCTION<m>[:FFT]:FREQUENCY:START? (see page 393)	<frequency> ::= the start frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:FREQUENCY:STOP <frequency> (see page 394)	:FUNCTION<m>[:FFT]:FREQUENCY:STOP? (see page 394)	<frequency> ::= the stop frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:GATE <gating> (see page 395)	:FUNCTION<m>[:FFT]:GATE? (see page 395)	<gating> ::= {NONE   ZOOM} <m> ::= 1-4 in NR1 format
:FUNCTION<m>[:FFT]:SPAN <span> (see page 396)	:FUNCTION<m>[:FFT]:SPAN? (see page 396)	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:VTPe <units> (see page 397)	:FUNCTION<m>[:FFT]:VTPe? (see page 397)	<units> ::= {DECibel   VRMS} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:WINDDOW <>window> (see page 398)	:FUNCTION<m>[:FFT]:WINDDOW? (see page 398)	<>window> ::= {RECTangular   HANNing   FLATtop   BHARris} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:FREQUENCY:HIGHpass <3dB_freq> (see page 399)	:FUNCTION<m>:FREQUENCY:HIGHpass? (see page 399)	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:FREQUENCY:LOWPass <3dB_freq> (see page 400)	:FUNCTION<m>:FREQUENCY:LOWPass? (see page 400)	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:INTEGRATE:IOFFset <input_offset> (see page 401)	:FUNCTION<m>:INTEGRATE:IOFFset? (see page 401)	<input_offset> ::= DC offset correction in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:LINEAR:GAIN <value> (see page 402)	:FUNCTION<m>:LINEAR:GAIN? (see page 402)	<value> ::= 'A' in Ax + B, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format

**Table 15** :FUNCtion<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCtion<m>:LINEar:OFFSet <value> (see page 403)	:FUNCtion<m>:LINEar:OFFSet? (see page 403)	<value> ::= 'B' in Ax + B, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCtion<m>:OFFSET <offset> (see page 404)	:FUNCtion<m>:OFFSET? (see page 404)	<offset> ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function. <m> ::= 1 to (# math functions) in NR1 format
:FUNCtion<m>:OPERation <operation> (see page 405)	:FUNCtion<m>:OPERation? (see page 407)	<operation> ::= {ADD   SUBTract   MULTiply   DIVide   INTegrate   DIFF   FFT   SQRT   MAGNify   ABSolute   SQUare   LN   LOG   EXP   TEN   LOWPass   HIGHpass   AVERage   LINEar   MAXimum   MINimum   PEAK   MAXHold   MINHold   TREnd   BTIMing   BSTate} <m> ::= 1 to (# math functions) in NR1 format
:FUNCtion<m>:RANGE <range> (see page 409)	:FUNCtion<m>:RANGE? (see page 409)	<range> ::= the full-scale vertical axis value in NR3 format. The range for ADD, SUBT, MULT is 8E-6 to 800E+3. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFF function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV. <m> ::= 1 to (# math functions) in NR1 format
:FUNCtion<m>:REFERenc e <level> (see page 410)	:FUNCtion<m>:REFERenc e? (see page 410)	<level> ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function. <m> ::= 1 to (# math functions) in NR1 format

**Table 15** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:SCALE <scale value>[<suffix>] (see <a href="#">page 411</a> )	:FUNCTION<m>:SCALE? (see <a href="#">page 411</a> )	<scale value> ::= integer in NR1 format <suffix> ::= {V   dB} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:SMOoth:P OINTs <points> (see <a href="#">page 412</a> )	:FUNCTION<m>:SMOoth:P OINTs? (see <a href="#">page 412</a> )	<points> ::= odd integer in NR1 format
:FUNCTION<m>:SOURcel <source> (see <a href="#">page 413</a> )	:FUNCTION<m>:SOURcel? (see <a href="#">page 413</a> )	<source> ::= {CHANnel<n>   FUNCTION<c>   MATH<c>   BUS<b>} <n> ::= 1 to (# analog channels) in NR1 format <c> ::= {1}, must be lower than <m> <b> ::= {1   2} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:SOURce2 <source> (see <a href="#">page 415</a> )	:FUNCTION<m>:SOURce2? (see <a href="#">page 415</a> )	<source> ::= {CHANnel<n>   NONE} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:TRENd:ME ASurement <type> (see <a href="#">page 416</a> )	:FUNCTION<m>:TRENd:ME ASurement? (see <a href="#">page 416</a> )	<type> ::= {VAverage   ACRMs   VRATio   PERiod   FREQuency   PWIDth   NWIDTH   DUTYcycle   RISetime   FALLtime} <m> ::= 1 to (# math functions) in NR1 format

**Table 16** :HARDcopy Commands Summary

Command	Query	Options and Query Returns
:HARDcopy:AREA <area> (see <a href="#">page 419</a> )	:HARDcopy:AREA? (see <a href="#">page 419</a> )	<area> ::= SCReen
:HARDcopy:APRinter <active_printer> (see <a href="#">page 420</a> )	:HARDcopy:APRinter? (see <a href="#">page 420</a> )	<active_printer> ::= {<index>   <name>} <index> ::= integer index of printer in list <name> ::= name of printer in list

**Table 16** :HARDcopy Commands Summary (continued)

Command	Query	Options and Query Returns
:HARDcopy:FACTors {{0   OFF}   {1   ON}} (see <a href="#">page 421</a> )	:HARDcopy:FACTors? (see <a href="#">page 421</a> )	{0   1}
:HARDcopy:FFEed {{0   OFF}   {1   ON}} (see <a href="#">page 422</a> )	:HARDcopy:FFEed? (see <a href="#">page 422</a> )	{0   1}
:HARDcopy:INKSaver {{0   OFF}   {1   ON}} (see <a href="#">page 423</a> )	:HARDcopy:INKSaver? (see <a href="#">page 423</a> )	{0   1}
:HARDcopy:LAYOUT <layout> (see <a href="#">page 424</a> )	:HARDcopy:LAYOUT? (see <a href="#">page 424</a> )	<layout> ::= {LANDscape   PORTRait}
:HARDcopy:NETWork:ADD Ress <address> (see <a href="#">page 425</a> )	:HARDcopy:NETWork:ADD Ress? (see <a href="#">page 425</a> )	<address> ::= quoted ASCII string
:HARDcopy:NETWork:APP Ly (see <a href="#">page 426</a> )	n/a	n/a
:HARDcopy:NETWork:DOM ain <domain> (see <a href="#">page 427</a> )	:HARDcopy:NETWork:DOM ain? (see <a href="#">page 427</a> )	<domain> ::= quoted ASCII string
:HARDcopy:NETWork:PAS Sword <password> (see <a href="#">page 428</a> )	n/a	<password> ::= quoted ASCII string
:HARDcopy:NETWork:SLOT <slot> (see <a href="#">page 429</a> )	:HARDcopy:NETWork:SLOT? (see <a href="#">page 429</a> )	<slot> ::= {NET0   NET1}
:HARDcopy:NETWork:USE Rname <username> (see <a href="#">page 430</a> )	:HARDcopy:NETWork:USE Rname? (see <a href="#">page 430</a> )	<username> ::= quoted ASCII string
:HARDcopy:PALETTE <palette> (see <a href="#">page 431</a> )	:HARDcopy:PALETTE? (see <a href="#">page 431</a> )	<palette> ::= {COLor   GRAYscale   NONE}

**Table 16** :HARDcopy Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:HARDcopy:PRINTER:LIS T? (see <a href="#">page 432</a> )	<list> ::= [<printer_spec>] ... <printer_spec> ::= "<index>,<active>,<name>;" <index> ::= integer index of printer <active> ::= {Y   N} <name> ::= name of printer
:HARDcopy:START (see <a href="#">page 433</a> )	n/a	n/a

**Table 17** :LISTer Commands Summary

Command	Query	Options and Query Returns
n/a	:LISTer:DATA? (see <a href="#">page 436</a> )	<binary_block> ::= comma-separated data with newlines at the end of each row
:LISTer:DISPlay {{OFF   0}   {SBUS1   ON   1}   {SBUS2   2}   ALL} (see <a href="#">page 437</a> )	:LISTer:DISPlay? (see <a href="#">page 437</a> )	{OFF   SBUS1   SBUS2   ALL}
:LISTer:REFerence <time_ref> (see <a href="#">page 438</a> )	:LISTer:REFerence? (see <a href="#">page 438</a> )	<time_ref> ::= {TRIGger   PREVIOUS}

**Table 18** :MARKer Commands Summary

Command	Query	Options and Query Returns
n/a	:MARKer:DYDX? (see <a href="#">page 442</a> )	<return_value> ::= •Y/•X value in NR3 format
:MARKer:MODE <mode> (see <a href="#">page 443</a> )	:MARKer:MODE? (see <a href="#">page 443</a> )	<mode> ::= {OFF   MEASurement   MANual   WAVEform   BINary   HEX}
:MARKer:X1:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 444</a> )	:MARKer:X1:DISPlay? (see <a href="#">page 444</a> )	<setting> ::= {0   1}
:MARKer:X1Position <position>[suffix] (see <a href="#">page 445</a> )	:MARKer:X1Position? (see <a href="#">page 445</a> )	<position> ::= X1 cursor position value in NR3 format [suffix] ::= {s   ms   us   ns   ps   Hz   kHz   MHz} <return_value> ::= X1 cursor position value in NR3 format

**Table 18** :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:X1Y1source <source> (see page 446)	:MARKer:X1Y1source? (see <a href="#">page 446</a> )	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= &lt;source&gt;</p>
:MARKer:X2:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 447</a> )	:MARKer:X2:DISPLAY? (see <a href="#">page 447</a> )	<setting> ::= {0   1}
:MARKer:X2Position <position>[suffix] (see <a href="#">page 448</a> )	:MARKer:X2Position? (see <a href="#">page 448</a> )	<p>&lt;position&gt; ::= X2 cursor position value in NR3 format</p> <p>[suffix] ::= {s   ms   us   ns   ps   Hz   kHz   MHz}</p> <p>&lt;return_value&gt; ::= X2 cursor position value in NR3 format</p>
:MARKer:X2Y2source <source> (see page 449)	:MARKer:X2Y2source? (see <a href="#">page 449</a> )	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= &lt;source&gt;</p>
n/a	:MARKer:XDELta? (see <a href="#">page 450</a> )	<return_value> ::= X cursors delta value in NR3 format
:MARKer:XUNits <mode> (see <a href="#">page 451</a> )	:MARKer:XUNits? (see <a href="#">page 451</a> )	<units> ::= {SEConds   HERTZ   DEGRees   PERCent}
:MARKer:XUNits:USE (see <a href="#">page 452</a> )	n/a	n/a
:MARKer:Y1:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 453</a> )	:MARKer:Y1:DISPLAY? (see <a href="#">page 453</a> )	<setting> ::= {0   1}

**Table 18** :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:Y1Position <position>[suffix] (see <a href="#">page 454</a> )	:MARKer:Y1Position? (see <a href="#">page 454</a> )	<position> ::= Y1 cursor position value in NR3 format [suffix] ::= {V   mV   dB} <return_value> ::= Y1 cursor position value in NR3 format
:MARKer:Y2:DISPLAY { {0   OFF}   {1   ON} } (see <a href="#">page 455</a> )	:MARKer:Y2:DISPLAY? (see <a href="#">page 455</a> )	<setting> ::= {0   1}
:MARKer:Y2Position <position>[suffix] (see <a href="#">page 456</a> )	:MARKer:Y2Position? (see <a href="#">page 456</a> )	<position> ::= Y2 cursor position value in NR3 format [suffix] ::= {V   mV   dB} <return_value> ::= Y2 cursor position value in NR3 format
n/a	:MARKer:YDELta? (see <a href="#">page 457</a> )	<return_value> ::= Y cursors delta value in NR3 format
:MARKer:YUNits <mode> (see <a href="#">page 458</a> )	:MARKer:YUNits? (see <a href="#">page 458</a> )	<units> ::= {BASE   PERCent}
:MARKer:YUNits:USE (see <a href="#">page 459</a> )	n/a	n/a

**Table 19** :MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:ALL (see <a href="#">page 478</a> )	n/a	n/a
:MEASure:AREA [<interval>] [, <source>] (see <a href="#">page 479</a> )	:MEASure:AREa? [<interval>] [, <source>] (see <a href="#">page 479</a> )	<interval> ::= {CYCLE   DISPLAY} <source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= area in volt-seconds, NR3 format

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:BRATE [<source>] (see page 480)	:MEASure:BRATE? [<source>] (see page 480)	<p>&lt;source&gt; ::= {&lt;digital channels&gt;   CHANNEL&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;}</p> <p>&lt;digital channels&gt; ::= DIGITAL&lt;d&gt; for the MSO models</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;n&gt; ::= 1 to (# of analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= bit rate in Hz, NR3 format</p>
:MEASure:BWIDth [<source>] (see page 481)	:MEASure:BWIDth? [<source>] (see page 481)	<p>&lt;source&gt; ::= {CHANNEL&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= burst width in seconds, NR3 format</p>
:MEASure:CLEar (see page 482)	n/a	n/a
:MEASure:COUNTER [<source>] (see page 483)	:MEASure:COUNTER? [<source>] (see page 483)	<p>&lt;source&gt; ::= {CHANNEL&lt;n&gt;   EXTERNAL} for DSO models</p> <p>&lt;source&gt; ::= {CHANNEL&lt;n&gt;   DIGITAL&lt;d&gt;   EXTERNAL} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= counter frequency in Hertz in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DEFine DElAy, <delay spec> (see <a href="#">page 485</a> )	:MEASure:DEFine? DElAy (see <a href="#">page 487</a> )	<delay spec> ::= <edge_spec1>,<edge_spec2> edge_spec1 ::= [<slope>]<occurrence> edge_spec2 ::= [<slope>]<occurrence> <slope> ::= {+   -} <occurrence> ::= integer
:MEASure:DEFine THRESHolds, <threshold spec> (see <a href="#">page 485</a> )	:MEASure:DEFine? THRESHolds (see <a href="#">page 487</a> )	<threshold spec> ::= {STANDARD}   {<threshold mode>,<upper>, <middle>,<lower>} <threshold mode> ::= {PERCENT   ABSOLUTE}
:MEASure:DElAy [<source1>] [,<source2>] (see <a href="#">page 488</a> )	:MEASure:DElAy? [<source1>] [,<source2>] (see <a href="#">page 488</a> )	<source1,2> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUAL:CHARge [<interval>] [,<source1>][,<source2>] (see <a href="#">page 490</a> )	:MEASure:DUAL:CHARge? [<interval>] [,<source1>][,<source2>] (see <a href="#">page 490</a> )	<interval> ::= {CYCLE   DISPLAY} <source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= area in Amp-hours, NR3 format
:MEASure:DUAL:VAMplitude [<source1>][,<source2>] (see <a href="#">page 491</a> )	:MEASure:DUAL:VAMplitude? [<source1>][,<source2>] (see <a href="#">page 491</a> )	<source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the amplitude of the selected waveform in volts in NR3 format

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DUAL:VAVerag e [<interval> [,,<source1>] [,,<source2>] (see <a href="#">page 492</a> )	:MEASure:DUAL:VAVerag e? [<interval> [,,<source1>] [,,<source2>] (see <a href="#">page 492</a> )	<interval> ::= {CYCLE   DISPLAY} <source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= calculated average voltage in NR3 format
:MEASure:DUAL:VBASe [<source1>] [,<source2>] (see <a href="#">page 493</a> )	:MEASure:DUAL:VBASe? [<source1>] [,<source2>] (see <a href="#">page 493</a> )	<source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <base_voltage> ::= voltage at the base of the selected waveform in NR3 format
:MEASure:DUAL:VPP [<source1>] [,<source2>] (see <a href="#">page 494</a> )	:MEASure:DUAL:VPP? [<source1>] [,<source2>] (see <a href="#">page 494</a> )	<source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= voltage peak-to-peak of the selected waveform in NR3 format
:MEASure:DUAL:VRMS [<interval>] [,<type>] [,,<source1>] [,,<source2>] (see <a href="#">page 495</a> )	:MEASure:DUAL:VRMS? [<interval>] [,<type>] [,,<source1>] [,,<source2>] (see <a href="#">page 495</a> )	<interval> ::= {CYCLE   DISPLAY} <type> ::= {AC   DC} <source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= calculated RMS voltage in NR3 format

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DUTYcycle [<source>] (see page 496)	:MEASure:DUTYcycle? [<source>] (see page 496)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= ratio of positive pulse width to period in NR3 format</p>
:MEASure:FALLtime [<source>] (see page 497)	:MEASure:FALLtime? [<source>] (see page 497)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= time in seconds between the lower and upper thresholds in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FREQuency [<source>] (see page 498)	:MEASure:FREQuency? [<source>] (see page 498)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= frequency in Hertz in NR3 format</p>
:MEASure:NDUTy [<source>] (see page 499)	:MEASure:NDUTy? [<source>] (see page 499)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= ratio of negative pulse width to period in NR3 format</p>
:MEASure:NEDGes [<source>] (see page 500)	:MEASure:NEDGes? [<source>] (see page 500)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the falling edge count in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NPULses [<source>] (see page 501)	:MEASure:NPULses? [<source>] (see page 501)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the falling pulse count in NR3 format</p>
:MEASure:NWIDth [<source>] (see page 502)	:MEASure:NWIDth? [<source>] (see page 502)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= negative pulse width in seconds-NR3 format</p>
:MEASure:OVERshoot [<source>] (see page 503)	:MEASure:OVERshoot? [<source>] (see page 503)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the percent of the overshoot of the selected waveform in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PEDGes [<source>] (see page 505)	:MEASure:PEDGes? [<source>] (see page 505)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the rising edge count in NR3 format</p>
:MEASure:PERiod [<source>] (see page 506)	:MEASure:PERiod? [<source>] (see page 506)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= waveform period in seconds in NR3 format</p>
:MEASure:PHASE [<source1>] [,<source2>] (see page 507)	:MEASure:PHASE? [<source1>] [,<source2>] (see page 507)	<p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the phase angle value in degrees in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PPULses [<source>] (see page 508)	:MEASure:PPULses? [<source>] (see page 508)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the rising pulse count in NR3 format</p>
:MEASure:PREShoot [<source>] (see page 509)	:MEASure:PREShoot? [<source>] (see page 509)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the percent of preshoot of the selected waveform in NR3 format</p>
:MEASure:PWIDth [<source>] (see page 510)	:MEASure:PWIDth? [<source>] (see page 510)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= width of positive pulse in seconds in NR3 format</p>
n/a	:MEASure:RESults? <result_list> (see page 511)	<p>&lt;result_list&gt; ::=</p> <p>comma-separated list of measurement results</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:RISetime [<source>] (see page 514)	:MEASure:RISetime? [<source>] (see page 514)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= rise time in seconds in NR3 format</p>
:MEASure:SDEViation [<source>] (see page 515)	:MEASure:SDEViation? [<source>] (see page 515)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= calculated std deviation in NR3 format</p>
:MEASure:SHOW {{0   OFF}   {1   ON}} (see page 516)	:MEASure:SHOW? (see page 516)	{0   1}
:MEASure:SOURce <source1> [,<source2>] (see page 517)	:MEASure:SOURce? (see page 517)	<p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;   EXTERNAL} for DSO models</p> <p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   DIGITAL&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMORY&lt;r&gt;   EXTERNAL} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= {&lt;source&gt;   NONE}</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:STATistics <type> (see <a href="#">page 519</a> )	:MEASure:STATistics? (see <a href="#">page 519</a> )	<type> ::= {{ON   1}   CURRent   MEAN   MINimum   MAXimum   STDDev   COUNT} ON ::= all statistics returned
:MEASure:STATistics:D ISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 520</a> )	:MEASure:STATistics:D ISPlay? (see <a href="#">page 520</a> )	{0   1}
:MEASure:STATistics:I NCREMENT (see <a href="#">page 521</a> )	n/a	n/a
:MEASure:STATistics:M CCount <setting> (see <a href="#">page 522</a> )	:MEASure:STATistics:M CCount? (see <a href="#">page 522</a> )	<setting> ::= {INFinite   <count>} <count> ::= 2 to 2000 in NR1 format
:MEASure:STATistics:R ESet (see <a href="#">page 523</a> )	n/a	n/a
:MEASure:STATistics:R SDeviation {{0   OFF}   {1   ON}} (see <a href="#">page 524</a> )	:MEASure:STATistics:R SDeviation? (see <a href="#">page 524</a> )	{0   1}
n/a	:MEASure:TEDGE? <slope><occurrence>[, <source>] (see <a href="#">page 525</a> )	<slope> ::= direction of the waveform <occurrence> ::= the transition to be reported <source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   FUNCTION<m>   MATH<m>   WMEMory<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= time in seconds of the specified transition

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:TValue? <value>, [<>slope>]<occurrence> [, <source>] (see page 527)	<p>&lt;value&gt; ::= voltage level that the waveform must cross.</p> <p>&lt;slope&gt; ::= direction of the waveform when &lt;value&gt; is crossed.</p> <p>&lt;occurrence&gt; ::= transitions reported.</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= time in seconds of specified voltage crossing in NR3 format</p>
:MEASure:VAMplitude [<source>] (see page 529)	:MEASure:VAMplitude? [<source>] (see page 529)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the amplitude of the selected waveform in volts in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAverage [<interval>] [,<source>] (see page 530)	:MEASure:VAverage? [<interval>] [,<source>] (see page 530)	<p>&lt;interval&gt; ::= {CYCLE   DISPLAY}</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= calculated average voltage in NR3 format</p>
:MEASure:VBASE [<source>] (see page 531)	:MEASure:VBASE? [<source>] (see page 531)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;base_voltage&gt; ::= voltage at the base of the selected waveform in NR3 format</p>
:MEASure:VMAX [<source>] (see page 532)	:MEASure:VMAX? [<source>] (see page 532)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= maximum voltage of the selected waveform in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VMIN [<source>] (see page 533)	:MEASure:VMIN? [<source>] (see page 533)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= minimum voltage of the selected waveform in NR3 format</p>
:MEASure:VPP [<source>] (see page 534)	:MEASure:VPP? [<source>] (see page 534)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= voltage peak-to-peak of the selected waveform in NR3 format</p>
:MEASure:VRATio [<interval>] [, <source 1>] [, <source2>] (see page 535)	:MEASure:VRATio? [<interval>] [, <source 1>] [, <source2>] (see page 535)	<p>&lt;interval&gt; ::= {CYCLE   DISPLAY}</p> <p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the ratio value in dB in NR3 format</p>

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VRMS [<interval>] [,<type>] [,<source>] (see <a href="#">page 536</a> )	:MEASure:VRMS? [<interval>] [,<type>] [,<source>] (see <a href="#">page 536</a> )	<interval> ::= {CYCLE   DISPLAY} <type> ::= {AC   DC} <source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= calculated dc RMS voltage in NR3 format
n/a	:MEASure:VTIMe? <vtimetime>[,<source>] (see <a href="#">page 537</a> )	<vtimetime> ::= displayed time from trigger in seconds in NR3 format <source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} for DSO models <source> ::= {CHANNEL<n>   DIGITAL<d>   FUNCTION<m>   MATH<m>   WMEMORY<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= voltage at the specified time in NR3 format
:MEASure:VTOP [<source>] (see <a href="#">page 538</a> )	:MEASure:VTOP? [<source>] (see <a href="#">page 538</a> )	<source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= voltage at the top of the waveform in NR3 format
:MEASure:WINDOW <type> (see <a href="#">page 539</a> )	:MEASure:WINDOW? (see <a href="#">page 539</a> )	<type> ::= {MAIN   ZOOM   AUTO   GATE}

**Table 19** :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:XMAX [<source>] (see page 540)	:MEASure:XMAX? [<source>] (see page 540)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   FFT   MATH&lt;m&gt;   WMMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</p>
:MEASure:XMIN [<source>] (see page 541)	:MEASure:XMIN? [<source>] (see page 541)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   FFT   MATH&lt;m&gt;   WMMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= horizontal value of the minimum in NR3 format</p>

**Table 20** :MEASure Power Commands Summary

Command	Query	Options and Query Returns
:MEASure:ANGLE [<source1>] [,<source2>] (see page 547)	:MEASure:ANGLE? [<source1>] [,<source2>] (see page 547)	<p>&lt;source1&gt;, &lt;source2&gt; ::= {CHANnel&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;return_value&gt; ::= the power phase angle in degrees in NR3 format</p>
:MEASure:APPARENT [<source1>] [,<source2>] (see page 548)	:MEASure:APPARENT? [<source1>] [,<source2>] (see page 548)	<p>&lt;source1&gt;, &lt;source2&gt; ::= {CHANnel&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;return_value&gt; ::= the apparent power value in NR3 format</p>

**Table 20** :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:CPLoss [<source1>] [,<source2>] (see page 549)	:MEASure:CPLoss? [<source1>] [,<source2>] (see page 549)	<source1>, <source2> <source1> ::= {FUNCTION<m>   MATH<m>} <source2> ::= {CHANnel<n>} <m> ::= 1 to (# math functions) in NR1 format <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the switching loss per cycle watts value in NR3 format
:MEASure:CRESt [<source>] (see page 550)	:MEASure:CRESt? [<source>] (see page 550)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= the crest factor value in NR3 format
:MEASure:EFFiciency (see page 551)	:MEASure:EFFiciency? (see page 551)	<return_value> ::= percent value in NR3 format
:MEASure:ELOSS [<source>] (see page 552)	:MEASure:ELOSS? [<source>] (see page 552)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the energy loss value in NR3 format
:MEASure:FACTOr [<source1>] [,<source2>] (see page 553)	:MEASure:FACTOr? [<source1>] [,<source2>] (see page 553)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the power factor value in NR3 format
:MEASure:IPOWer (see page 554)	:MEASure:IPOWer? (see page 554)	<return_value> ::= the input power value in NR3 format

**Table 20** :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:OFFTime [<source1>] [,<source2>] (see <a href="#">page 555</a> )	:MEASure:OFFTime? [<source1>] [,<source2>] (see <a href="#">page 555</a> )	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the time in seconds in NR3 format
:MEASure:ONTime [<source1>] [,<source2>] (see <a href="#">page 556</a> )	:MEASure:ONTime? [<source1>] [,<source2>] (see <a href="#">page 556</a> )	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the time in seconds in NR3 format
:MEASure:OPOWer (see <a href="#">page 557</a> )	:MEASure:OPOWer? (see <a href="#">page 557</a> )	<return_value> ::= the output power value in NR3 format
:MEASure:PCURrent [<source>] (see <a href="#">page 558</a> )	:MEASure:PCURrent? [<source>] (see <a href="#">page 558</a> )	<source> ::= {CHANnel<n>   FUNCTion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the peak current value in NR3 format
:MEASure:PLOSS [<source>] (see <a href="#">page 559</a> )	:MEASure:PLOSS? [<source>] (see <a href="#">page 559</a> )	<source> ::= {CHANnel<n>   FUNCTion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the power loss value in NR3 format

**Table 20** :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:RDSon [<source1> [, <source2>] (see <a href="#">page 560</a> )	:MEASure:RDSon? [<source1> [, <source2>] (see <a href="#">page 560</a> )	<source1>, <source2> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the VCE(sat) value in NR3 format
:MEASure:REACTive [<source1> [, <source2>] (see <a href="#">page 561</a> )	:MEASure:REACTive? [<source1> [, <source2>] (see <a href="#">page 561</a> )	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the reactive power value in NR3 format
:MEASure:REAL [<source>] (see <a href="#">page 562</a> )	:MEASure:REAL? [<source>] (see <a href="#">page 562</a> )	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= the real power value in NR3 format
:MEASure:RIPPLE [<source>] (see <a href="#">page 563</a> )	:MEASure:RIPPLE? [<source>] (see <a href="#">page 563</a> )	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the output ripple value in NR3 format

**Table 20** :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TRESPonse [ <source/> ] (see page 564)	:MEASure:TRESPonse? [ <source/> ] (see page 564)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= time in seconds for the overshoot to settle back into the band in NR3 format</p>
:MEASure:VCESat [ <source/> ] (see page 565)	:MEASure:VCESat? [ <source/> ] (see page 565)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the VCE(sat) value in NR3 format</p>

**Table 21** :MTEST Commands Summary

Command	Query	Options and Query Returns
:MTEST:ALL {{0   OFF}   {1   ON}} (see page 572)	:MTEST:ALL? (see page 572)	{0   1}
:MTEST:AMASK:CREATE (see page 573)	n/a	n/a
:MTEST:AMASK:SOURce <source> (see page 574)	:MTEST:AMASK:SOURce? (see page 574)	<p>&lt;source&gt; ::= CHANnel&lt;n&gt;</p> <p>&lt;n&gt; ::= {1   2   3   4} for 4ch models</p> <p>&lt;n&gt; ::= {1   2} for 2ch models</p>
:MTEST:AMASK:UNITS <units> (see page 575)	:MTEST:AMASK:UNITS? (see page 575)	<units> ::= {CURREnt   DIVisions}
:MTEST:AMASK:XDELta <value> (see page 576)	:MTEST:AMASK:XDELta? (see page 576)	<value> ::= X delta value in NR3 format

**Table 21** :MTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:MTEST:AMASK:YDELta <value> (see page 577)	:MTEST:AMASK:YDELta? (see page 577)	<value> ::= Y delta value in NR3 format
n/a	:MTEST:COUNT:FWAVEforms? [CHANnel<n>] (see page 578)	<failed> ::= number of failed waveforms in NR1 format
:MTEST:COUNT:RESet (see page 579)	n/a	n/a
n/a	:MTEST:COUNT:TIME? (see page 580)	<time> ::= elapsed seconds in NR3 format
n/a	:MTEST:COUNT:WAVeform s? (see page 581)	<count> ::= number of waveforms in NR1 format
:MTEST:DATA <mask> (see page 582)	:MTEST:DATA? (see page 582)	<mask> ::= data in IEEE 488.2 # format.
:MTEST:DELETE (see page 583)	n/a	n/a
:MTEST:ENABLE {{0   OFF}   {1   ON}} (see page 584)	:MTEST:ENABLE? (see page 584)	{0   1}
:MTEST:LOCK {{0   OFF}   {1   ON}} (see page 585)	:MTEST:LOCK? (see page 585)	{0   1}
:MTEST:RMODE <rmode> (see page 586)	:MTEST:RMODE? (see page 586)	<rmode> ::= {FORever   TIME   SIGMa   WAveforms}
:MTEST:RMODE:FACTion: MEASure {{0   OFF}   {1   ON}} (see page 587)	:MTEST:RMODE:FACTion: MEASure? (see page 587)	{0   1}
:MTEST:RMODE:FACTion: PRINT {{0   OFF}   {1   ON}} (see page 588)	:MTEST:RMODE:FACTion: PRINT? (see page 588)	{0   1}
:MTEST:RMODE:FACTion: SAVE {{0   OFF}   {1   ON}} (see page 589)	:MTEST:RMODE:FACTion: SAVE? (see page 589)	{0   1}
:MTEST:RMODE:FACTion: STOP {{0   OFF}   {1   ON}} (see page 590)	:MTEST:RMODE:FACTion: STOP? (see page 590)	{0   1}
:MTEST:RMODE:SIGMa <level> (see page 591)	:MTEST:RMODE:SIGMa? (see page 591)	<level> ::= from 0.1 to 9.3 in NR3 format

**Table 21** :MTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:MTEST:RMODE:TIME <seconds> (see <a href="#">page 592</a> )	:MTEST:RMODE:TIME? (see <a href="#">page 592</a> )	<seconds> ::= from 1 to 86400 in NR3 format
:MTEST:RMODE:WAVeform s <count> (see <a href="#">page 593</a> )	:MTEST:RMODE:WAVeform s? (see <a href="#">page 593</a> )	<count> ::= number of waveforms in NR1 format
:MTEST:SCALe:BIND {{0   OFF}   {1   ON}} (see <a href="#">page 594</a> )	:MTEST:SCALe:BIND? (see <a href="#">page 594</a> )	{0   1}
:MTEST:SCALe:X1 <x1_value> (see <a href="#">page 595</a> )	:MTEST:SCALe:X1? (see <a href="#">page 595</a> )	<x1_value> ::= X1 value in NR3 format
:MTEST:SCALe:XDELta <xdelta_value> (see <a href="#">page 596</a> )	:MTEST:SCALe:XDELta? (see <a href="#">page 596</a> )	<xdelta_value> ::= X delta value in NR3 format
:MTEST:SCALe:Y1 <y1_value> (see <a href="#">page 597</a> )	:MTEST:SCALe:Y1? (see <a href="#">page 597</a> )	<y1_value> ::= Y1 value in NR3 format
:MTEST:SCALe:Y2 <y2_value> (see <a href="#">page 598</a> )	:MTEST:SCALe:Y2? (see <a href="#">page 598</a> )	<y2_value> ::= Y2 value in NR3 format
:MTEST:SOURce <source> (see <a href="#">page 599</a> )	:MTEST:SOURce? (see <a href="#">page 599</a> )	<source> ::= {CHANnel<n>   NONE} <n> ::= {1   2   3   4} for 4ch models <n> ::= {1   2} for 2ch models
n/a	:MTEST:TITLe? (see <a href="#">page 600</a> )	<title> ::= a string of up to 128 ASCII characters

**Table 22** :POD<n> Commands Summary

Command	Query	Options and Query Returns
:POD<n>:DISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 603</a> )	:POD<n>:DISPlay? (see <a href="#">page 603</a> )	{0   1} <n> ::= 1-2 in NR1 format

**Table 22** :POD<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:POD<n>:SIZE <value> (see <a href="#">page 604</a> )	:POD<n>:SIZE? (see <a href="#">page 604</a> )	<value> ::= {SMALL   MEDIUM   LARGe}
:POD<n>:THreshold <type>[suffix] (see <a href="#">page 605</a> )	:POD<n>:THreshold? (see <a href="#">page 605</a> )	<n> ::= 1-2 in NR1 format <type> ::= {CMOS   ECL   TTL   <user defined value>} <user defined value> ::= value in NR3 format [suffix] ::= {V   mV   uV }

**Table 23** :POWER Commands Summary

Command	Query	Options and Query Returns
:POWER:CLResponse:APP Ly (see <a href="#">page 613</a> )	n/a	n/a
:POWER:CLResponse:FRE Quency:START <value>[suffix] (see <a href="#">page 614</a> )	:POWER:CLResponse:FRE Quency:START? (see <a href="#">page 614</a> )	<value> ::= {20   100   1000   10000   100000   1000000   10000000} [suffix] ::= {Hz   kHz   MHz}
:POWER:CLResponse:FRE Quency:STOP <value>[suffix] (see <a href="#">page 615</a> )	:POWER:CLResponse:FRE Quency:STOP? (see <a href="#">page 615</a> )	<value> ::= {100   1000   10000   100000   1000000   10000000   20000000} [suffix] ::= {Hz   kHz   MHz}
:POWER:CLResponse:VIEW <view_type> (see <a href="#">page 616</a> )	:POWER:CLResponse:VIEW? (see <a href="#">page 616</a> )	<view_type> ::= {GAIN   PHASE}
:POWER:CLResponse:YMAXimum <value> (see <a href="#">page 617</a> )	:POWER:CLResponse:YMAXimum? (see <a href="#">page 617</a> )	<value> ::= dB value in multiples of 10 from -110 to 120
:POWER:CLResponse:YMINimum <value> (see <a href="#">page 618</a> )	:POWER:CLResponse:YMINimum? (see <a href="#">page 618</a> )	<value> ::= dB value in multiples of 10 from -120 to 110
:POWER:DESKew (see <a href="#">page 619</a> )	n/a	n/a
:POWER:EFFiciency:APP Ly (see <a href="#">page 620</a> )	n/a	n/a
:POWER:EFFiciency:TYPE <type> (see <a href="#">page 621</a> )	:POWER:EFFiciency:TYPE? (see <a href="#">page 621</a> )	<type> ::= {DCDC   DCAC   ACDC   ACAC}

**Table 23** :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:ENABLE {{0   OFF}   {1   ON}} (see page 622)	:POWer:ENABLE? (see page 622)	{0   1}
:POWer:HARMonics:APPLy (see page 623)	n/a	n/a
n/a	:POWer:HARMonics:DATA? (see page 624)	<binary_block> ::= comma-separated data with newlines at the end of each row
:POWer:HARMonics:DISPLAY <display> (see page 625)	:POWer:HARMonics:DISPLAY? (see page 625)	<display> ::= {TABLE   BAR   OFF}
n/a	:POWer:HARMonics:FAILCOUNT? (see page 626)	<count> ::= integer in NR1 format
:POWer:HARMonics:LINE <frequency> (see page 627)	:POWer:HARMonics:LINE? (see page 627)	<frequency> ::= {F50   F60   F400}
n/a	:POWer:HARMonics:POWERFACtor? (see page 628)	<value> ::= Class C power factor in NR3 format
:POWer:HARMonics:RPOWER <source> (see page 629)	:POWer:HARMonics:RPOWER? (see page 629)	<source> ::= {MEASured   USER}
:POWer:HARMonics:RPOWER:USER <value> (see page 630)	:POWer:HARMonics:RPOWER:USER? (see page 630)	<value> ::= Watts from 1.0 to 600.0 in NR3 format
n/a	:POWer:HARMonics:RUNCOUNT? (see page 631)	<count> ::= integer in NR1 format
:POWer:HARMonics:STANDARD <class> (see page 632)	:POWer:HARMonics:STANDARD? (see page 632)	<class> ::= {A   B   C   D}
n/a	:POWer:HARMonics:STATUS? (see page 633)	<status> ::= {PASS   FAIL   UNTESTED}
n/a	:POWer:HARMonics:THD? (see page 634)	<value> ::= Total Harmonics Distortion in NR3 format
:POWer:INRUSH:APPLY (see page 635)	n/a	n/a
:POWer:INRUSH:EXIT (see page 636)	n/a	n/a

**Table 23** :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:INRush:NEXT (see <a href="#">page 637</a> )	n/a	n/a
:POWer:MODulation:APP Ly (see <a href="#">page 638</a> )	n/a	n/a
:POWer:MODulation:SOU Rce <source> (see <a href="#">page 639</a> )	:POWer:MODulation:SOU Rce? (see <a href="#">page 639</a> )	<source> ::= {V   I}
:POWer:MODulation:TYP E <modulation> (see <a href="#">page 640</a> )	:POWer:MODulation:TYP E? (see <a href="#">page 640</a> )	<modulation> ::= {VAverage   ACRMs   VRATio   PERiod   FREQuency   PWIDith   NWIDth   DUTYcycle   RISetime   FALLtime}
:POWer:ONOFF:APPLY (see <a href="#">page 641</a> )	n/a	n/a
:POWer:ONOFF:EXIT (see <a href="#">page 642</a> )	n/a	n/a
:POWer:ONOFF:NEXT (see <a href="#">page 643</a> )	n/a	n/a
:POWer:ONOFF:TEST {{0   OFF}   {1   ON}} (see <a href="#">page 644</a> )	:POWer:ONOFF:TEST? (see <a href="#">page 644</a> )	{0   1}
:POWer:PSRR:APPLy (see <a href="#">page 645</a> )	n/a	n/a
:POWer:PSRR:FREQuency :MAXimum <value>[suffix] (see <a href="#">page 646</a> )	:POWer:PSRR:FREQuency :MAXimum? (see <a href="#">page 646</a> )	<value> ::= {10   100   1000   10000   100000   1000000   10000000   20000000} [suffix] ::= {Hz   kHz   MHz}
:POWer:PSRR:FREQuency :MINimum <value>[suffix] (see <a href="#">page 647</a> )	:POWer:PSRR:FREQuency :MINimum? (see <a href="#">page 647</a> )	<value> ::= {1   10   100   1000   10000   100000   1000000   10000000} [suffix] ::= {Hz   kHz   MHz}
:POWer:PSRR:RMAXimum <value> (see <a href="#">page 648</a> )	:POWer:PSRR:RMAXimum? (see <a href="#">page 648</a> )	<value> ::= Maximum ratio value in NR1 format
:POWer:QUALity:APPLy (see <a href="#">page 649</a> )	n/a	n/a
:POWer:RIPPLE:APPLy (see <a href="#">page 650</a> )	n/a	n/a
:POWer:SIGNals:AUTose tup <analysis> (see <a href="#">page 651</a> )	n/a	<analysis> ::= {HARMonics   EFFiciency   RIPPLE   MODulation   QUALity   SLEW   SWITch   RDSVce}

**Table 23** :POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:SIGNals:CYCLes :HARMonics <count> (see <a href="#">page 652</a> )	:POWER:SIGNals:CYCLes :HARMonics? (see <a href="#">page 652</a> )	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWER:SIGNals:CYCLes :QUALity <count> (see <a href="#">page 653</a> )	:POWER:SIGNals:CYCLes :QUALity? (see <a href="#">page 653</a> )	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWER:SIGNals:DURati on:EFFiciency <value>[suffix] (see <a href="#">page 654</a> )	:POWER:SIGNals:DURati on:EFFiciency? (see <a href="#">page 654</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:MODulation <value>[suffix] (see <a href="#">page 655</a> )	:POWER:SIGNals:DURati on:MODulation? (see <a href="#">page 655</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:ONOFF:OFF <value>[suffix] (see <a href="#">page 656</a> )	:POWER:SIGNals:DURati on:ONOFF:OFF? (see <a href="#">page 656</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:ONOFF:ON <value>[suffix] (see <a href="#">page 657</a> )	:POWER:SIGNals:DURati on:ONOFF:ON? (see <a href="#">page 657</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:RIPPle <value>[suffix] (see <a href="#">page 658</a> )	:POWER:SIGNals:DURati on:RIPPle? (see <a href="#">page 658</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:TRANSient <value>[suffix] (see <a href="#">page 659</a> )	:POWER:SIGNals:DURati on:TRANSient? (see <a href="#">page 659</a> )	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:IEXPec ted <value>[suffix] (see <a href="#">page 660</a> )	:POWER:SIGNals:IEXPec ted? (see <a href="#">page 660</a> )	<value> ::= Expected current value in NR3 format [suffix] ::= {A   mA}
:POWER:SIGNals:OVERsh oot <percent> (see <a href="#">page 661</a> )	:POWER:SIGNals:OVERsh oot? (see <a href="#">page 661</a> )	<percent> ::= percent of overshoot value in NR1 format [suffix] ::= {V   mV}}
:POWER:SIGNals:VMAXim um:INRush <value>[suffix] (see <a href="#">page 662</a> )	:POWER:SIGNals:VMAXim um:INRush? (see <a href="#">page 662</a> )	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}

**Table 23** :POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:SIGNals:VMAXimum:ONOFF:OFF <value>[suffix] (see page 663)	:POWER:SIGNals:VMAXimum:ONOFF:OFF? (see page 663)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VMAXimum:ONOFF:ON <value>[suffix] (see page 664)	:POWER:SIGNals:VMAXimum:ONOFF:ON? (see page 664)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:ONOFF:OFF <value>[suffix] (see page 665)	:POWER:SIGNals:VSTeady:ONOFF:OFF? (see page 665)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:ONOFF:ON <value>[suffix] (see page 666)	:POWER:SIGNals:VSTeady:ONOFF:ON? (see page 666)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:TRANSient <value>[suffix] (see page 667)	:POWER:SIGNals:VSTeady:TRANSient? (see page 667)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:SOURce:CURREnt<i> <source> (see page 668)	:POWER:SIGNals:SOURce:CURREnt<i>? (see page 668)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:SIGNals:SOURce:VOLTage<i> <source> (see page 669)	:POWER:SIGNals:SOURce:VOLTage<i>? (see page 669)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:SLEW:APPLy (see page 670)	n/a	n/a
:POWER:SLEW:SOURce <source> (see page 671)	:POWER:SLEW:SOURce? (see page 671)	<source> ::= {V   I}
:POWER:SWITch:APPLy (see page 672)	n/a	n/a
:POWER:SWITch:CONDUCTion <conduction> (see page 673)	:POWER:SWITch:CONDUCTion? (see page 673)	<conduction> ::= {WAVEform   RDS   VCE}
:POWER:SWITch:IREFerence <percent> (see page 674)	:POWER:SWITch:IREFerence? (see page 674)	<percent> ::= percent in NR1 format

**Table 23** :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:SWITch:RDS <value>[suffix] (see page 675)	:POWer:SWITch:RDS? (see page 675)	<value> ::= Rds(on) value in NR3 format [suffix] ::= {OHM   mOHM}
:POWer:SWITch:VCE <value>[suffix] (see page 676)	:POWer:SWITch:VCE? (see page 676)	<value> ::= Vce(sat) value in NR3 format [suffix] ::= {V   mV}
:POWer:SWITch:VREFerence <percent> (see page 677)	:POWer:SWITch:VREFerence? (see page 677)	<percent> ::= percent in NR1 format
:POWer:TRANSient:APPLy (see page 678)	n/a	n/a
:POWer:TRANSient:EXIT (see page 679)	n/a	n/a
:POWer:TRANSient:IINITial <value>[suffix] (see page 680)	:POWer:TRANSient:IINITial? (see page 680)	<value> ::= Initial current value in NR3 format [suffix] ::= {A   mA}
:POWer:TRANSient:INEW <value>[suffix] (see page 681)	:POWer:TRANSient:INEW? (see page 681)	<value> ::= New current value in NR3 format [suffix] ::= {A   mA}
:POWer:TRANSient:NEXT (see page 682)	n/a	n/a

**Table 24** :RECall Commands Summary

Command	Query	Options and Query Returns
:RECall:ARbitrary[:START] [<file_spec>] [, <column>] [, <wavegen_id>] (see page 685)	n/a	<p>&lt;file_spec&gt; ::= {&lt;internal_loc&gt;   &lt;file_name&gt;}</p> <p>&lt;column&gt; ::= Column in CSV file to load. Column number starts from 1.</p> <p>&lt;internal_loc&gt; ::= 0-3; an integer in NR1 format</p> <p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>&lt;wavegen_id&gt; ::= WGEN1</p>
:RECall:DBC[:START] [<file_name>] [, <serialbus>] (see page 686)	n/a	<p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>If extension included in file name, it must be ".dbc".</p> <p>&lt;serialbus&gt; ::= {SBUS&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# of serial bus) in NR1 format</p>
:RECall:FILEname <base_name> (see page 687)	:RECall:FILEname? (see page 687)	<p>&lt;base_name&gt; ::= quoted ASCII string</p>
:RECall:LDF[:START] [<file_name>] [, <serialbus>] (see page 688)	n/a	<p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>If extension included in file name, it must be ".ldf".</p> <p>&lt;serialbus&gt; ::= {SBUS&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# of serial bus) in NR1 format</p>
:RECall:MASK[:START] [<file_spec>] (see page 689)	n/a	<p>&lt;file_spec&gt; ::= {&lt;internal_loc&gt;   &lt;file_name&gt;}</p> <p>&lt;internal_loc&gt; ::= 0-3; an integer in NR1 format</p> <p>&lt;file_name&gt; ::= quoted ASCII string</p>
:RECall:PWD <path_name> (see page 690)	:RECall:PWD? (see page 690)	<p>&lt;path_name&gt; ::= quoted ASCII string</p>

**Table 24** :RECall Commands Summary (continued)

Command	Query	Options and Query Returns
:RECall:SETup [:START] [<file_spec>] (see page 691)	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:RECall:WMEMory<r>[:START] [<file_name>] (see page 692)	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format <file_name> ::= quoted ASCII string If extension included in file name, it must be ".h5".

**Table 25** :SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:ARBitrary[:STARt] [<file_spec>] [, <wavegen_id>] (see page 697)	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-3; an integer in NR1 format <file_name> ::= quoted ASCII string <wavegen_id> ::= WGEN1
:SAVE:FILEname <base_name> (see page 698)	:SAVE:FILEname? (see page 698)	<base_name> ::= quoted ASCII string
:SAVE:IMAGE[:START] [<file_name>] (see page 699)	n/a	<file_name> ::= quoted ASCII string
:SAVE:IMAGE:FACTors {{0   OFF}   {1   ON}} (see page 700)	:SAVE:IMAGE:FACTors? (see page 700)	{0   1}
:SAVE:IMAGE:FORMAT <format> (see page 701)	:SAVE:IMAGE:FORMAT? (see page 701)	<format> ::= {{BMP   BMP24bit}   BMP8bit   PNG   NONE}
:SAVE:IMAGE:INKSaver {{0   OFF}   {1   ON}} (see page 702)	:SAVE:IMAGE:INKSaver? (see page 702)	{0   1}
:SAVE:IMAGE:PALETTE <palette> (see page 703)	:SAVE:IMAGE:PALETTE? (see page 703)	<palette> ::= {COLOR   GRAYscale}

**Table 25** :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:LISTER[:START] [<file_name>] (see <a href="#">page 704</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:MASK[:START] [<file_spec>] (see <a href="#">page 705</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-3; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:MULTi[:START] [<file_name>] (see <a href="#">page 706</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:POWer[:START] [<file_name>] (see <a href="#">page 707</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:PWD <path_name> (see <a href="#">page 708</a> )	:SAVE:PWD? (see <a href="#">page 708</a> )	<path_name> ::= quoted ASCII string
:SAVE:RESults[:START] [<file_spec>] (see <a href="#">page 709</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:RESults:FORMAT: CURSOR {{0   OFF}   {1   ON}} (see <a href="#">page 710</a> )	:SAVE:RESults:FORMAT: CURSOR? (see <a href="#">page 710</a> )	{0   1}
:SAVE:RESults:FORMAT: MASK {{0   OFF}   {1   ON}} (see <a href="#">page 711</a> )	:SAVE:RESults:FORMAT: MASK? (see <a href="#">page 711</a> )	{0   1}
:SAVE:RESults:FORMAT: MEASurement {{0   OFF}   {1   ON}} (see <a href="#">page 712</a> )	:SAVE:RESults:FORMAT: MEASurement? (see <a href="#">page 712</a> )	{0   1}
:SAVE:RESults:FORMAT: SEARch {{0   OFF}   {1   ON}} (see <a href="#">page 713</a> )	:SAVE:RESults:FORMAT: SEARch? (see <a href="#">page 713</a> )	{0   1}
:SAVE:RESults:FORMAT: SEGmented {{0   OFF}   {1   ON}} (see <a href="#">page 714</a> )	:SAVE:RESults:FORMAT: SEGmented? (see <a href="#">page 714</a> )	{0   1}

**Table 25** :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:SETup[:START] [<file_spec>] (see <a href="#">page 715</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:WAVeform[:STARt] [<file_name>] (see <a href="#">page 716</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:WAVeform:FORMAT <format> (see <a href="#">page 717</a> )	:SAVE:WAVeform:FORMAT ? (see <a href="#">page 717</a> )	<format> ::= {ASCIixy   CSV   BINary   NONE}
:SAVE:WAVeform:LENGTH <length> (see <a href="#">page 718</a> )	:SAVE:WAVeform:LENGTH ? (see <a href="#">page 718</a> )	<length> ::= 100 to max. length; an integer in NR1 format
:SAVE:WAVeform:LENGTH :MAX {{0   OFF}   {1   ON}} (see <a href="#">page 719</a> )	:SAVE:WAVeform:LENGTH :MAX? (see <a href="#">page 719</a> )	{0   1}
:SAVE:WAVeform:SEGMen ted <option> (see <a href="#">page 720</a> )	:SAVE:WAVeform:SEGMen ted? (see <a href="#">page 720</a> )	<option> ::= {ALL   CURRent}
:SAVE:WMEMory:SOURce <source> (see <a href="#">page 721</a> )	:SAVE:WMEMory:SOURce? (see <a href="#">page 721</a> )	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format NOTE: Only ADD or SUBtract math operations can be saved as reference waveforms. <return_value> ::= <source>
:SAVE:WMEMory[:STARt] [<file_name>] (see <a href="#">page 722</a> )	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".h5".

**Table 26** General :SBUS<n> Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:DISPlay {{0   OFF}   {1   ON}} (see page 726)	:SBUS<n>:DISPlay? (see page 726)	{0   1}
:SBUS<n>:MODE <mode> (see page 727)	:SBUS<n>:MODE? (see page 727)	<mode> ::= {A429   CAN   FLEXray   I2S   IIC   LIN   M1553   SENT   SPI   UART}

**Table 27** :SBUS<n>:A429 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:A429:AUTOset up (see page 730)	n/a	n/a
:SBUS<n>:A429:BASE <base> (see page 731)	:SBUS<n>:A429:BASE? (see page 731)	<base> ::= {BINary   HEX}
n/a	:SBUS<n>:A429:COUNT:ERRor? (see page 732)	<error_count> ::= integer in NR1 format
:SBUS<n>:A429:COUNT:RESet (see page 733)	n/a	n/a
n/a	:SBUS<n>:A429:COUNT:WORD? (see page 734)	<word_count> ::= integer in NR1 format
:SBUS<n>:A429:FORMAT <format> (see page 735)	:SBUS<n>:A429:FORMAT? (see page 735)	<format> ::= {LDSDi   LDSSm   LDATA}
:SBUS<n>:A429:SIGNAl <signal> (see page 736)	:SBUS<n>:A429:SIGNAl? (see page 736)	<signal> ::= {A   B   DIFFerential}
:SBUS<n>:A429:SOURce <source> (see page 737)	:SBUS<n>:A429:SOURce? (see page 737)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:A429:SPEEd <speed> (see page 738)	:SBUS<n>:A429:SPEEd? (see page 738)	<speed> ::= {LOW   HIGH}

**Table 27** :SBUS<n>:A429 Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:A429:TRIGger :LABel <value> (see page 739)	:SBUS<n>:A429:TRIGger :LABel? (see page 739)	<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 or "0xXX" (don't care) <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}
:SBUS<n>:A429:TRIGger :PATTern:DATA <string> (see page 740)	:SBUS<n>:A429:TRIGger :PATTern:DATA? (see page 740)	<string> ::= "nn...n" where n ::= {0   1   X}, length depends on FORMat
:SBUS<n>:A429:TRIGger :PATTern:SDI <string> (see page 741)	:SBUS<n>:A429:TRIGger :PATTern:SDI? (see page 741)	<string> ::= "nn" where n ::= {0   1   X}, length always 2 bits
:SBUS<n>:A429:TRIGger :PATTern:SSM <string> (see page 742)	:SBUS<n>:A429:TRIGger :PATTern:SSM? (see page 742)	<string> ::= "nn" where n ::= {0   1   X}, length always 2 bits
:SBUS<n>:A429:TRIGger :RANGE <min>,<max> (see page 743)	:SBUS<n>:A429:TRIGger :RANGE? (see page 743)	<min> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <max> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}
:SBUS<n>:A429:TRIGger :TYPE <condition> (see page 744)	:SBUS<n>:A429:TRIGger :TYPE? (see page 744)	<condition> ::= {WSTArt   WSTOp   LABel   LBITS   PERRor   WERRor   GERRor   WGERRors   ALLerrors   LRANge   ABITs   AOBits   AZBits}

**Table 28** :SBUS<n>:CAN Commands Summary

Command	Query	Options and Query Returns
n/a	:SBUS<n>:CAN:COUNT:ER Ror? (see <a href="#">page 748</a> )	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:OV ERload? (see <a href="#">page 749</a> )	<frame_count> ::= 0 in NR1 format
:SBUS<n>:CAN:COUNT:RE Set (see <a href="#">page 750</a> )	n/a	n/a
n/a	:SBUS<n>:CAN:COUNT:SP EC? (see <a href="#">page 751</a> )	<spec_error_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:TO Tal? (see <a href="#">page 752</a> )	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:UT ILization? (see <a href="#">page 753</a> )	<percent> ::= floating-point in NR3 format
:SBUS<n>:CAN:DISPLAY <type> (see <a href="#">page 754</a> )	:SBUS<n>:CAN:DISPLAY? (see <a href="#">page 754</a> )	<type> ::= {HEXAdecimal   SYMBOLic}
:SBUS<n>:CAN:FDSPoint <value> (see <a href="#">page 755</a> )	:SBUS<n>:CAN:FDSPoint ? (see <a href="#">page 755</a> )	<value> ::= even numbered percentages from 30 to 90 in NR3 format.
:SBUS<n>:CAN:FDSTanda rd <std> (see <a href="#">page 756</a> )	:SBUS<n>:CAN:FDSTanda rd? (see <a href="#">page 756</a> )	<std> ::= {ISO   NISO}
:SBUS<n>:CAN:SAMPLEpo int <percent> (see <a href="#">page 757</a> )	:SBUS<n>:CAN:SAMPLEpo int? (see <a href="#">page 757</a> )	<percent> ::= 30.0 to 90.0 in NR3 format
:SBUS<n>:CAN:SIGNAl:B AUDrate <baudrate> (see <a href="#">page 758</a> )	:SBUS<n>:CAN:SIGNAl:B AUDrate? (see <a href="#">page 758</a> )	<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments, or 5000000
:SBUS<n>:CAN:SIGNAl:D EFinition <value> (see <a href="#">page 759</a> )	:SBUS<n>:CAN:SIGNAl:D EFinition? (see <a href="#">page 759</a> )	<value> ::= {CANH   CANL   RX   TX   DIFFerential   DIFL   DIFH}
:SBUS<n>:CAN:SIGNAl:F DBaudrate <baudrate> (see <a href="#">page 760</a> )	:SBUS<n>:CAN:SIGNAl:F DBaudrate? (see <a href="#">page 760</a> )	<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.

**Table 28** :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:SOURCE <source> (see page 761)	:SBUS<n>:CAN:SOURce? (see page 761)	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   } for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:CAN:TRIGger <condition> (see page 762)	:SBUS<n>:CAN:TRIGger? (see page 763)	<condition> ::= {SOF   EOF   IDData   DATA   FDData   IDRmote   IDEither   ERRor   ACKerror   FORMrror   STUFFrror   CRCrror   SPECrror   ALLerrors   BRSBit   CRCDBit   EBActive   EBPassive   OVERload   MESSage   MSIGnal   FDMSignal}
:SBUS<n>:CAN:TRIGger: IDFilter {{0   OFF}   {1   ON}} (see page 765)	:SBUS<n>:CAN:TRIGger: IDFilter? (see page 765)	{0   1}
:SBUS<n>:CAN:TRIGger: PATtern:DATA <string> (see page 766)	:SBUS<n>:CAN:TRIGger: PATtern:DATA? (see page 766)	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:CAN:TRIGger: PATtern:DATA:DLC <dlc> (see page 767)	:SBUS<n>:CAN:TRIGger: PATtern:DATA:DLC? (see page 767)	<dlc> ::= integer between -1 (don't care) and 64, in NR1 format.
:SBUS<n>:CAN:TRIGger: PATtern:DATA:LENGTH <length> (see page 768)	:SBUS<n>:CAN:TRIGger: PATtern:DATA:LENGTH? (see page 768)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:CAN:TRIGger: PATtern:DATA:START <start> (see page 769)	:SBUS<n>:CAN:TRIGger: PATtern:DATA:START? (see page 769)	<start> ::= integer between 0 and 63, in NR1 format.
:SBUS<n>:CAN:TRIGger: PATtern:ID <string> (see page 770)	:SBUS<n>:CAN:TRIGger: PATtern:ID? (see page 770)	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:CAN:TRIGger: PATtern:ID:MODE <value> (see page 771)	:SBUS<n>:CAN:TRIGger: PATtern:ID:MODE? (see page 771)	<value> ::= {STANDARD   EXTENDED}

**Table 28** :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:TRIGger: SYMBolic:MESSage <name> (see page 772)	:SBUS<n>:CAN:TRIGger: SYMBolic:MESSage? (see page 772)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger: SYMBolic:SIGNal <name> (see page 773)	:SBUS<n>:CAN:TRIGger: SYMBolic:SIGNal? (see page 773)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger: SYMBolic:VALue <data> (see page 774)	:SBUS<n>:CAN:TRIGger: SYMBolic:VALue? (see page 774)	<data> ::= value in NR3 format

**Table 29** :SBUS<n>:FLEXray Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:FLEXray:AUTO setup (see page 777)	n/a	n/a
:SBUS<n>:FLEXray:BAUD rate <baudrate> (see page 778)	:SBUS<n>:FLEXray:BAUD rate? (see page 778)	<baudrate> ::= {2500000   5000000   10000000}
:SBUS<n>:FLEXray:CHAN nel <channel> (see page 779)	:SBUS<n>:FLEXray:CHAN nel? (see page 779)	<channel> ::= {A   B}
n/a	:SBUS<n>:FLEXray:COUN t:NULL? (see page 780)	<frame_count> ::= integer in NR1 format
:SBUS<n>:FLEXray:COUN t:RESET (see page 781)	n/a	n/a
n/a	:SBUS<n>:FLEXray:COUN t:SYNC? (see page 782)	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:FLEXray:COUN t:TOTAl? (see page 783)	<frame_count> ::= integer in NR1 format
:SBUS<n>:FLEXray:SOUR ce <source> (see page 784)	:SBUS<n>:FLEXray:SOUR ce? (see page 784)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format
:SBUS<n>:FLEXray:TRIG ger <condition> (see page 785)	:SBUS<n>:FLEXray:TRIG ger? (see page 785)	<condition> ::= {FRAMe   ERRor   EVENT}
:SBUS<n>:FLEXray:TRIG ger:ERRor:TYPE <error_type> (see page 786)	:SBUS<n>:FLEXray:TRIG ger:ERRor:TYPE? (see page 786)	<error_type> ::= {ALL   HCRC   FCRC}

**Table 29** :SBUS<n>:FLEXray Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:FLEXray:TRIG ger:EVENT:AUTOset (see page 787)	n/a	n/a
:SBUS<n>:FLEXray:TRIG ger:EVENT:BSS:ID <frame_id> (see page 788)	:SBUS<n>:FLEXray:TRIG ger:EVENT:BSS:ID? (see page 788)	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047
:SBUS<n>:FLEXray:TRIG ger:EVENT:TYPE <event> (see page 789)	:SBUS<n>:FLEXray:TRIG ger:EVENT:TYPE? (see page 789)	<event> ::= {WAKEup   TSS   {FES   DTS}   BSS}
:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCBase <cycle_count_base> (see page 790)	:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCBase? (see page 790)	<cycle_count_base> ::= integer from 0-63
:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCRepetitio n <cycle_count_repetiti on> (see page 791)	:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCRepetitio n? (see page 791)	<cycle_count_repetition> ::= {ALL   <rep #>} <rep #> ::= integer values 2, 4, 8, 16, 32, or 64
:SBUS<n>:FLEXray:TRIG ger:FRAMe:ID <frame_id> (see page 792)	:SBUS<n>:FLEXray:TRIG ger:FRAMe:ID? (see page 792)	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047
:SBUS<n>:FLEXray:TRIG ger:FRAMe:TYPE <frame_type> (see page 793)	:SBUS<n>:FLEXray:TRIG ger:FRAMe:TYPE? (see page 793)	<frame_type> ::= {NORMal   STARtup   NULL   SYNC   NSTArtup   NNULL   NSYNC   ALL}

**Table 30** :SBUS<n>:I2S Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:I2S:ALIGNmen t <setting> (see page 796)	:SBUS<n>:I2S:ALIGNmen t? (see page 796)	<setting> ::= {I2S   LJ   RJ}
:SBUS<n>:I2S:BASE <base> (see page 797)	:SBUS<n>:I2S:BASE? (see page 797)	<base> ::= {DECimal   HEX}
:SBUS<n>:I2S:CLOCK:SL OPe <slope> (see page 798)	:SBUS<n>:I2S:CLOCK:SL OPe? (see page 798)	<slope> ::= {NEGative   POSitive}

**Table 30** :SBUS<n>:I2S Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:I2S:RWIDth <receiver> (see <a href="#">page 799</a> )	:SBUS<n>:I2S:RWIDth? (see <a href="#">page 799</a> )	<receiver> ::= 4-32 in NR1 format
:SBUS<n>:I2S:SOURce:C LOCK <source> (see <a href="#">page 800</a> )	:SBUS<n>:I2S:SOURce:C LOCK? (see <a href="#">page 800</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:I2S:SOURce:D ATA <source> (see <a href="#">page 801</a> )	:SBUS<n>:I2S:SOURce:D ATA? (see <a href="#">page 801</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:I2S:SOURce:W SElect <source> (see <a href="#">page 802</a> )	:SBUS<n>:I2S:SOURce:W SElect? (see <a href="#">page 802</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:I2S:TRIGger <operator> (see <a href="#">page 803</a> )	:SBUS<n>:I2S:TRIGger? (see <a href="#">page 803</a> )	<operator> ::= {EQUAL   NOTEqual   LESSthan   GREaterthan   INRange   OUTRange   INCREasing   DECReasing}
:SBUS<n>:I2S:TRIGger: AUDIO <audio_ch> (see <a href="#">page 805</a> )	:SBUS<n>:I2S:TRIGger: AUDIO? (see <a href="#">page 805</a> )	<audio_ch> ::= {RIGHT   LEFT   EITHER}
:SBUS<n>:I2S:TRIGger: PATTern:DATA <string> (see <a href="#">page 806</a> )	:SBUS<n>:I2S:TRIGger: PATTern:DATA? (see <a href="#">page 807</a> )	<string> ::= "n" where n ::= 32-bit integer in signed decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   X   \$} when <base> = BINary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$} when <base> = HEX

**Table 30** :SBUS<n>:I2S Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:I2S:TRIGger: PATTern:FORMAT <base> (see <a href="#">page 808</a> )	:SBUS<n>:I2S:TRIGger: PATTern:FORMAT? (see <a href="#">page 808</a> )	<base> ::= {BINary   HEX   DECimal}
:SBUS<n>:I2S:TRIGger: RANGE <lower>,<upper> (see <a href="#">page 809</a> )	:SBUS<n>:I2S:TRIGger: RANGE? (see <a href="#">page 809</a> )	<lower> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <upper> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal
:SBUS<n>:I2S:TWIDth <word_size> (see <a href="#">page 811</a> )	:SBUS<n>:I2S:TWIDth? (see <a href="#">page 811</a> )	<word_size> ::= 4-32 in NR1 format
:SBUS<n>:I2S:WSLow <low_def> (see <a href="#">page 812</a> )	:SBUS<n>:I2S:WSLow? (see <a href="#">page 812</a> )	<low_def> ::= {LEFT   RIGHT}

**Table 31** :SBUS<n>:IIC Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:IIC:ASIZE <size> (see <a href="#">page 814</a> )	:SBUS<n>:IIC:ASIZE? (see <a href="#">page 814</a> )	<size> ::= {BIT7   BIT8}
:SBUS<n>:IIC[:SOURce] :CLOCK <source> (see <a href="#">page 815</a> )	:SBUS<n>:IIC[:SOURce] :CLOCK? (see <a href="#">page 815</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 31** :SBUS<n>:IIC Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:IIC[:SOURce] :DATA <source> (see page 816)	:SBUS<n>:IIC[:SOURce] :DATA? (see page 816)	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:IIC:TRIGger:PATTern:ADDRess <value> (see page 817)	:SBUS<n>:IIC:TRIGger:PATTern:ADDRess? (see page 817)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger:PATTern:DATA <value> (see page 818)	:SBUS<n>:IIC:TRIGger:PATTern:DATA? (see page 818)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger:PATTern:DATa2 <value> (see page 819)	:SBUS<n>:IIC:TRIGger:PATTern:DATa2? (see page 819)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger:QUALifier <value> (see page 820)	:SBUS<n>:IIC:TRIGger:QUALifier? (see page 820)	<value> ::= {EQUAL   NOTequal   LESSthan   GREaterthan}
:SBUS<n>:IIC:TRIGger[:TYPE] <type> (see page 821)	:SBUS<n>:IIC:TRIGger[:TYPE]? (see page 821)	<type> ::= {START   STOP   READ7   READEprom   WRITE7   WRITE10   NACKnowledge   ANACK   R7Data2   W7Data2   REStart}

**Table 32** :SBUS<n>:LIN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:LIN:DISPLAY <type> (see page 825)	:SBUS<n>:LIN:DISPLAY? (see page 825)	<type> ::= {HEXAdecimal   SYMBOLic}
:SBUS<n>:LIN:PARity {{0   OFF}   {1   ON}} (see page 826)	:SBUS<n>:LIN:PARity? (see page 826)	{0   1}
:SBUS<n>:LIN:SAMPLEpo int <value> (see page 827)	:SBUS<n>:LIN:SAMPLEpo int? (see page 827)	<value> ::= {60   62.5   68   70   75   80   87.5} in NR3 format
:SBUS<n>:LIN:SIGNAl:B AUDrate <baudrate> (see page 828)	:SBUS<n>:LIN:SIGNAl:B AUDrate? (see page 828)	<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments

**Table 32** :SBUS<n>:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:LIN:SOURce <source> (see page 829)	:SBUS<n>:LIN:SOURce? (see page 829)	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:LIN:STANDARD <std> (see page 830)	:SBUS<n>:LIN:STANDARD? (see page 830)	<std> ::= {LIN13   LIN20}
:SBUS<n>:LIN:SYNCbreak <value> (see page 831)	:SBUS<n>:LIN:SYNCbreak? (see page 831)	<value> ::= integer = {11   12   13}
:SBUS<n>:LIN:TRIGger <condition> (see page 832)	:SBUS<n>:LIN:TRIGger? (see page 832)	<condition> ::= {SYNCbreak   ID   DATA}
:SBUS<n>:LIN:TRIGger: ID <value> (see page 833)	:SBUS<n>:LIN:TRIGger: ID? (see page 833)	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9   A,...,F} for hexadecimal
:SBUS<n>:LIN:TRIGger: PATtern:DATA <string> (see page 834)	:SBUS<n>:LIN:TRIGger: PATtern:DATA? (see page 834)	<string> ::= "n" where n ::= 32-bit integer in unsigned decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   X   \$} when <base> = BINary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$} when <base> = HEX
:SBUS<n>:LIN:TRIGger: PATtern:DATA:LENGTH <length> (see page 836)	:SBUS<n>:LIN:TRIGger: PATtern:DATA:LENGTH? (see page 836)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:LIN:TRIGger: PATtern:FORMAT <base> (see page 837)	:SBUS<n>:LIN:TRIGger: PATtern:FORMAT? (see page 837)	<base> ::= {BINary   HEX   DECimal}

**Table 32** :SBUS<n>:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:LIN:TRIGger: SYMBolic:FRAMe <name> (see page 838)	:SBUS<n>:LIN:TRIGger: SYMBolic:FRAMe? (see page 838)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger: SYMBolic:SIGNal <name> (see page 839)	:SBUS<n>:LIN:TRIGger: SYMBolic:SIGNal? (see page 839)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger: SYMBolic:VALue <data> (see page 840)	:SBUS<n>:LIN:TRIGger: SYMBolic:VALue? (see page 840)	<data> ::= value in NR3 format

**Table 33** :SBUS<n>:M1553 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:M1553:AUTose tup (see page 842)	n/a	n/a
:SBUS<n>:M1553:BASE <base> (see page 843)	:SBUS<n>:M1553:BASE? (see page 843)	<base> ::= {BINary   HEX}
:SBUS<n>:M1553:SOURce <source> (see page 844)	:SBUS<n>:M1553:SOURce? (see page 844)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:M1553:TRIGge r:PATTern:DATA <string> (see page 845)	:SBUS<n>:M1553:TRIGge r:PATTern:DATA? (see page 845)	<string> ::= "nn...n" where n ::= {0   1   X}
:SBUS<n>:M1553:TRIGge r:RTA <value> (see page 846)	:SBUS<n>:M1553:TRIGge r:RTA? (see page 846)	<value> ::= 5-bit integer in decimal, <nondecimal>, or <string> from 0-31 <nondecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F}
:SBUS<n>:M1553:TRIGge r:TYPE <type> (see page 847)	:SBUS<n>:M1553:TRIGge r:TYPE? (see page 847)	<type> ::= {DSTArt   DSTOp   CSTArt   CSTOp   RTA   PERRor   SERRor   MERRor   RTA11}

**Table 34** :SBUS<n>:SENT Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SENT:CLOCK <period> (see page 851)	:SBUS<n>:SENT:CLOCK? (see page 851)	<period> ::= the nominal clock period (tick), from 1 us to 300 us, in NR3 format.
:SBUS<n>:SENT:CRC <format> (see page 852)	:SBUS<n>:SENT:CRC? (see page 852)	<format> ::= {LEGacy   RECommended}
:SBUS<n>:SENT:DISPlay <base> (see page 853)	:SBUS<n>:SENT:DISPlay? (see page 853)	<base> ::= {HEX   DECimal   SYMBolic}
:SBUS<n>:SENT:FORMAT <decode> (see page 855)	:SBUS<n>:SENT:FORMAT? (see page 855)	<decode> ::= {NIBBles   FSIGnal   FSSerial   FESerial   SSERial   ESErial}
:SBUS<n>:SENT:IDLE <state> (see page 857)	:SBUS<n>:SENT:IDLE? (see page 857)	<state> ::= {LOW   HIGH}
:SBUS<n>:SENT:LENGTH <#_nibbles> (see page 858)	:SBUS<n>:SENT:LENGTH? (see page 858)	<#_nibbles> ::= from 1-6, in NR1 format.
:SBUS<n>:SENT:PPULse { {0   OFF}   {1   ON} } (see page 859)	:SBUS<n>:SENT:PPULse? (see page 859)	{0   1}
:SBUS<n>:SENT:SIGNAl<s>:DISPlay { {0   OFF}   {1   ON} } (see page 860)	:SBUS<n>:SENT:SIGNAl<s>:DISPlay? (see page 860)	<s> ::= 1-6, in NR1 format. {0   1}
:SBUS<n>:SENT:SIGNAl<s>:LENGTH <length> (see page 861)	:SBUS<n>:SENT:SIGNAl<s>:LENGTH? (see page 861)	<s> ::= 1-6, in NR1 format. <length> ::= from 1-24, in NR1 format.
:SBUS<n>:SENT:SIGNAl<s>:MULTiplier <multiplier> (see page 863)	:SBUS<n>:SENT:SIGNAl<s>:MULTiplier? (see page 863)	<s> ::= 1-6, in NR1 format. <multiplier> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNAl<s>:OFFSet <offset> (see page 864)	:SBUS<n>:SENT:SIGNAl<s>:OFFSet? (see page 864)	<s> ::= 1-6, in NR1 format. <offset> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNAl<s>:ORDer <order> (see page 865)	:SBUS<n>:SENT:SIGNAl<s>:ORDer? (see page 865)	<s> ::= 1-6, in NR1 format. <order> ::= {MSNFirst   LSNFirst}
:SBUS<n>:SENT:SIGNAl<s>:START <position> (see page 867)	:SBUS<n>:SENT:SIGNAl<s>:START? (see page 867)	<s> ::= 1-6, in NR1 format. <position> ::= from 0-23, in NR1 format.

**Table 34** :SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:SOURce <source> (see <a href="#">page 869</a> )	:SBUS<n>:SENT:SOURce? (see <a href="#">page 869</a> )	<source> ::= {CHANnel<n>   DIGItal<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SENT:TOLERance <percent> (see <a href="#">page 871</a> )	:SBUS<n>:SENT:TOLERance? (see <a href="#">page 871</a> )	<percent> ::= from 3-30, in NR1 format.
:SBUS<n>:SENT:TRIGger <mode> (see <a href="#">page 872</a> )	:SBUS<n>:SENT:TRIGger? (see <a href="#">page 872</a> )	<mode> ::= {SFCMessage   SSCMesage   FCData   SCMid   SCData   FCCerror   SCCerror   CRCerror   TOLerror   PPERror   SSPerror}
:SBUS<n>:SENT:TRIGger :FAST:DATA <string> (see <a href="#">page 874</a> )	:SBUS<n>:SENT:TRIGger :FAST:DATA? (see <a href="#">page 874</a> )	<string> ::= "nnnn..." where n ::= {0   1   X} <string> ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SENT:TRIGger :SLOW:DATA <data> (see <a href="#">page 875</a> )	:SBUS<n>:SENT:TRIGger :SLOW:DATA? (see <a href="#">page 875</a> )	<data> ::= when ILength = SHORT, from -1 (don't care) to 65535, in NR1 format. <data> ::= when ILength = LONG, from -1 (don't care) to 4095, in NR1 format.
:SBUS<n>:SENT:TRIGger :SLOW:ID <id> (see <a href="#">page 877</a> )	:SBUS<n>:SENT:TRIGger :SLOW:ID? (see <a href="#">page 877</a> )	<id> ::= when ILength = SHORT, from -1 (don't care) to 15, in NR1 format. <id> ::= when ILength = LONG, from -1 (don't care) to 255, in NR1 format.
:SBUS<n>:SENT:TRIGger :SLOW:ILENghth <length> (see <a href="#">page 879</a> )	:SBUS<n>:SENT:TRIGger :SLOW:ILENghth? (see <a href="#">page 879</a> )	<length> ::= {SHORT   LONG}
:SBUS<n>:SENT:TRIGger :TOLerance <percent> (see <a href="#">page 880</a> )	:SBUS<n>:SENT:TRIGger :TOLerance? (see <a href="#">page 880</a> )	<percent> ::= from 1-18, in NR1 format.

**Table 35** :SBUS<n>:SPI Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SPI:BITorder <order> (see <a href="#">page 883</a> )	:SBUS<n>:SPI:BITorder ? (see <a href="#">page 883</a> )	<order> ::= {LSBFFirst   MSBFFirst}
:SBUS<n>:SPI:CLOCK:SOOPe <slope> (see <a href="#">page 884</a> )	:SBUS<n>:SPI:CLOCK:SOOPe? (see <a href="#">page 884</a> )	<slope> ::= {NEGative   POSitive}
:SBUS<n>:SPI:CLOCK:TIMEout <time_value> (see <a href="#">page 885</a> )	:SBUS<n>:SPI:CLOCK:TIMEout? (see <a href="#">page 885</a> )	<time_value> ::= time in seconds in NR3 format
:SBUS<n>:SPI:FRAMing <value> (see <a href="#">page 886</a> )	:SBUS<n>:SPI:FRAMing? (see <a href="#">page 886</a> )	<value> ::= {CHIPselect   {NCHipselect   NOTC}   TIMEout}
:SBUS<n>:SPI:SOURce:LOCK <source> (see <a href="#">page 887</a> )	:SBUS<n>:SPI:SOURce:LOCK? (see <a href="#">page 887</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGital<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:FRAME <source> (see <a href="#">page 888</a> )	:SBUS<n>:SPI:SOURce:FRAME? (see <a href="#">page 888</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGital<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:ISO <source> (see <a href="#">page 889</a> )	:SBUS<n>:SPI:SOURce:ISO? (see <a href="#">page 889</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGital<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:MOXI <source> (see <a href="#">page 890</a> )	:SBUS<n>:SPI:SOURce:MOXI? (see <a href="#">page 890</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGital<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 35** :SBUS<n>:SPI Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SPI:TRIGger: PATtern:MISO:DATA <string> (see <a href="#">page 891</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MISO:DATA? (see <a href="#">page 891</a> )	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SPI:TRIGger: PATtern:MISO:WIDTH <width> (see <a href="#">page 892</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MISO:WIDTH? (see <a href="#">page 892</a> )	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger: PATtern:MOsi:DATA <string> (see <a href="#">page 893</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MOsi:DATA? (see <a href="#">page 893</a> )	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SPI:TRIGger: PATtern:MOsi:WIDTH <width> (see <a href="#">page 894</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MOsi:WIDTH? (see <a href="#">page 894</a> )	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger: TYPE <value> (see <a href="#">page 895</a> )	:SBUS<n>:SPI:TRIGger: TYPE? (see <a href="#">page 895</a> )	<value> ::= {MOsi   MISO}
:SBUS<n>:SPI:WIDTH <word_width> (see <a href="#">page 896</a> )	:SBUS<n>:SPI:WIDTH? (see <a href="#">page 896</a> )	<word_width> ::= integer 4-16 in NR1 format

**Table 36** :SBUS<n>:UART Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:UART:BASE <base> (see <a href="#">page 900</a> )	:SBUS<n>:UART:BASE? (see <a href="#">page 900</a> )	<base> ::= {ASCii   BINary   HEX}
:SBUS<n>:UART:BAUDrate <baudrate> (see <a href="#">page 901</a> )	:SBUS<n>:UART:BAUDrate? (see <a href="#">page 901</a> )	<baudrate> ::= integer from 100 to 8000000
:SBUS<n>:UART:BITorder <bitorder> (see <a href="#">page 902</a> )	:SBUS<n>:UART:BITorder? (see <a href="#">page 902</a> )	<bitorder> ::= {LSBFirst   MSBFFirst}
n/a	:SBUS<n>:UART:COUNT:EROR? (see <a href="#">page 903</a> )	<frame_count> ::= integer in NR1 format
:SBUS<n>:UART:COUNT:RESet (see <a href="#">page 904</a> )	n/a	n/a
n/a	:SBUS<n>:UART:COUNT:RXFRAMES? (see <a href="#">page 905</a> )	<frame_count> ::= integer in NR1 format

**Table 36** :SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:SBUS<n>:UART:COUNT:T XFRAMES? (see <a href="#">page 906</a> )	<frame_count> ::= integer in NR1 format
:SBUS<n>:UART:FRAMing <value> (see <a href="#">page 907</a> )	:SBUS<n>:UART:FRAMing ? (see <a href="#">page 907</a> )	<value> ::= {OFF   <decimal>   <nondecimal>} <decimal> ::= 8-bit integer from 0-255 (0x00-0xff) <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary
:SBUS<n>:UART:PARity <parity> (see <a href="#">page 908</a> )	:SBUS<n>:UART:PARity? (see <a href="#">page 908</a> )	<parity> ::= {EVEN   ODD   NONE}
:SBUS<n>:UART:POLarit y <polarity> (see <a href="#">page 909</a> )	:SBUS<n>:UART:POLarit y? (see <a href="#">page 909</a> )	<polarity> ::= {HIGH   LOW}
:SBUS<n>:UART:SOURce: RX <source> (see <a href="#">page 910</a> )	:SBUS<n>:UART:SOURce: RX? (see <a href="#">page 910</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:UART:SOURce: TX <source> (see <a href="#">page 911</a> )	:SBUS<n>:UART:SOURce: TX? (see <a href="#">page 911</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:UART:TRIGger :BASE <base> (see <a href="#">page 912</a> )	:SBUS<n>:UART:TRIGger :BASE? (see <a href="#">page 912</a> )	<base> ::= {ASCII   HEX}
:SBUS<n>:UART:TRIGger :BURSt <value> (see <a href="#">page 913</a> )	:SBUS<n>:UART:TRIGger :BURSt? (see <a href="#">page 913</a> )	<value> ::= {OFF   1 to 4096 in NR1 format}

**Table 36** :SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:UART:TRIGger :DATA <value> (see page 914)	:SBUS<n>:UART:TRIGger :DATA? (see page 914)	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, <binary>, or <quoted_string> format <hexadecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0   1} for binary <quoted_string> ::= any of the 128 valid 7-bit ASCII characters (or standard abbreviations)
:SBUS<n>:UART:TRIGger :IDLE <time_value> (see page 915)	:SBUS<n>:UART:TRIGger :IDLE? (see page 915)	<time_value> ::= time from 1 us to 10 s in NR3 format
:SBUS<n>:UART:TRIGger :QUALifier <value> (see page 916)	:SBUS<n>:UART:TRIGger :QUALifier? (see page 916)	<value> ::= {EQUal   NOTequal   GREaterthan   LESSthan}
:SBUS<n>:UART:TRIGger :TYPE <value> (see page 917)	:SBUS<n>:UART:TRIGger :TYPE? (see page 917)	<value> ::= {RSTArt   RSTOP   RDATa   RD1   RD0   RDx   PARityerror   TSTArt   TSTOP   TDATa   TD1   TD0   TDx}
:SBUS<n>:UART:WIDTH <width> (see page 918)	:SBUS<n>:UART:WIDTH? (see page 918)	<width> ::= {5   6   7   8   9}

**Table 37** General :SEARch Commands Summary

Command	Query	Options and Query Returns
n/a	:SEARch:COUNT? (see page 921)	<count> ::= an integer count value
:SEARch:EVENT <event_number> (see page 922)	:SEARch:EVENT? (see page 922)	<event_number> ::= the integer number of a found search event
:SEARch:MODE <value> (see page 923)	:SEARch:MODE? (see page 923)	<value> ::= {EDGE   GLITch   RUNT   TRANSition   SERial{1   2}   PEAK}
:SEARch:STATE <value> (see page 924)	:SEARch:STATE? (see page 924)	<value> ::= {{0   OFF}   {1   ON}}

**Table 38** :SEARch:EDGE Commands Summary

Command	Query	Options and Query Returns
:SEARch:EDGE:SLOPe <slope> (see <a href="#">page 926</a> )	:SEARch:EDGE:SLOPe? (see <a href="#">page 926</a> )	<slope> ::= {POSiTive   NEGative   EITHer}
:SEARch:EDGE:SOURce <source> (see <a href="#">page 927</a> )	:SEARch:EDGE:SOURce? (see <a href="#">page 927</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

**Table 39** :SEARch:GLITch Commands Summary

Command	Query	Options and Query Returns
:SEARch:GLITch:GREAtre rthan <greater_than_time>[s uffix] (see <a href="#">page 929</a> )	:SEARch:GLITch:GREAtre rthan? (see <a href="#">page 929</a> )	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:LESSthan <less_than_time>[suff ix] (see <a href="#">page 930</a> )	:SEARch:GLITch:LESSthan? (see <a href="#">page 930</a> )	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:POLarity <polarity> (see <a href="#">page 931</a> )	:SEARch:GLITch:POLarity? (see <a href="#">page 931</a> )	<polarity> ::= {POSiTive   NEGative}
:SEARch:GLITch:QUALif ier <qualifier> (see <a href="#">page 932</a> )	:SEARch:GLITch:QUALif ier? (see <a href="#">page 932</a> )	<qualifier> ::= {GREAterthan   LESSthan   RANGE}
:SEARch:GLITch:RANGE <less_than_time>[suff ix], <greater_than_time>[s uffix] (see <a href="#">page 933</a> )	:SEARch:GLITch:RANGE? (see <a href="#">page 933</a> )	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:SOURce <source> (see <a href="#">page 934</a> )	:SEARch:GLITch:SOURce? (see <a href="#">page 934</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

**Table 40** :SEARch:PEAK Commands Summary

Command	Query	Options and Query Returns
:SEARch:PEAK:EXCursion <delta_level> (see page 936)	:SEARch:PEAK:EXCursion? (see <a href="#">page 936</a> )	<delta_level> ::= required change in level to be recognized as a peak, in NR3 format.
:SEARch:PEAK:NPeaks <number> (see page 937)	:SEARch:PEAK:NPeaks? (see <a href="#">page 937</a> )	<number> ::= max number of peaks to find, 1-11 in NR1 format.
:SEARch:PEAK:SOURce <source> (see page 938)	:SEARch:PEAK:SOURce? (see <a href="#">page 938</a> )	<source> ::= {FUNCTION<m>   MATH<m>} (must be an FFT waveform) <m> ::= 1 to 4 in NR1 format
:SEARch:PEAK:THReShold <level> (see page 939)	:SEARch:PEAK:THReShold? (see <a href="#">page 939</a> )	<level> ::= necessary level to be considered a peak, in NR3 format.

**Table 41** :SEARch:RUNT Commands Summary

Command	Query	Options and Query Returns
:SEARch:RUNT:POLarity <polarity> (see page 941)	:SEARch:RUNT:POLarity? (see <a href="#">page 941</a> )	<polarity> ::= {POSitive   NEGative   EITHer}
:SEARch:RUNT:QUALifier <qualifier> (see page 942)	:SEARch:RUNT:QUALifier? (see <a href="#">page 942</a> )	<qualifier> ::= {GREaterthan   LESSthan   NONE}
:SEARch:RUNT:SOURce <source> (see page 943)	:SEARch:RUNT:SOURce? (see <a href="#">page 943</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARch:RUNT:TIME <time>[suffix] (see page 944)	:SEARch:RUNT:TIME? (see <a href="#">page 944</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 42** :SEARch:TRANSition Commands Summary

Command	Query	Options and Query Returns
:SEARch:TRANSition:QUALifier <qualifier> (see page 946)	:SEARch:TRANSition:QUALifier? (see <a href="#">page 946</a> )	<qualifier> ::= {GREaterthan   LESSthan}
:SEARch:TRANSition:SLOPe <slope> (see page 947)	:SEARch:TRANSition:SLOPe? (see <a href="#">page 947</a> )	<slope> ::= {NEGative   POSitive}

**Table 42** :SEARch:TRANSition Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARch:TRANSition:SO URce <source> (see <a href="#">page 948</a> )	:SEARch:TRANSition:SO URce? (see <a href="#">page 948</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARch:TRANSition:TI ME <time>[suffix] (see <a href="#">page 949</a> )	:SEARch:TRANSition:TI ME? (see <a href="#">page 949</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 43** :SEARch:SERial:A429 Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:A429:L ABel <value> (see <a href="#">page 951</a> )	:SEARch:SERial:A429:L ABel? (see <a href="#">page 951</a> )	<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}
:SEARch:SERial:A429:M ODE <condition> (see <a href="#">page 952</a> )	:SEARch:SERial:A429:M ODE? (see <a href="#">page 952</a> )	<condition> ::= {LABEL   LBITS   PERRor   WERRor   GERRor   WGERrors   ALLerrors}
:SEARch:SERial:A429:P ATTern:DATA <string> (see <a href="#">page 953</a> )	:SEARch:SERial:A429:P ATTern:DATA? (see <a href="#">page 953</a> )	<string> ::= "nn...n" where n ::= {0   1}, length depends on FORMat
:SEARch:SERial:A429:P ATTern:SDI <string> (see <a href="#">page 954</a> )	:SEARch:SERial:A429:P ATTern:SDI? (see <a href="#">page 954</a> )	<string> ::= "nn" where n ::= {0   1}, length always 2 bits
:SEARch:SERial:A429:P ATTern:SSM <string> (see <a href="#">page 955</a> )	:SEARch:SERial:A429:P ATTern:SSM? (see <a href="#">page 955</a> )	<string> ::= "nn" where n ::= {0   1}, length always 2 bits

**Table 44** :SEARch:SERial:CAN Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:CAN:MODE <value> (see page 957)	:SEARch:SERial:CAN:MODE? (see page 957)	<value> ::= {IDData   DATA   IDRremote   IDEither   ERRor   ACKerror   FORMrror   STUFFerror   CRCerror   ALLerrors   OVERload   MESSage   MSIGnal}
:SEARch:SERial:CAN:PA TTern:DATA <string> (see page 959)	:SEARch:SERial:CAN:PA TTern:DATA? (see page 959)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} for hexadecimal
:SEARch:SERial:CAN:PA TTern:DATA:LENGTH <length> (see page 960)	:SEARch:SERial:CAN:PA TTern:DATA:LENGTH? (see page 960)	<length> ::= integer from 1 to 8 in NR1 format
:SEARch:SERial:CAN:PA TTern:ID <string> (see page 961)	:SEARch:SERial:CAN:PA TTern:ID? (see page 961)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} for hexadecimal
:SEARch:SERial:CAN:PA TTern:ID:MODE <value> (see page 962)	:SEARch:SERial:CAN:PA TTern:ID:MODE? (see page 962)	<value> ::= {STANDARD   EXTENDED}
:SEARch:SERial:CAN:SY MBolic:MESSAge <name> (see page 963)	:SEARch:SERial:CAN:SY MBolic:MESSAge? (see page 963)	<name> ::= quoted ASCII string
:SEARch:SERial:CAN:SY MBolic:SIGNAl <name> (see page 964)	:SEARch:SERial:CAN:SY MBolic:SIGNAl? (see page 964)	<name> ::= quoted ASCII string
:SEARch:SERial:CAN:SY MBolic:VALue <data> (see page 965)	:SEARch:SERial:CAN:SY MBolic:VALue? (see page 965)	<data> ::= value in NR3 format

**Table 45** :SEARch:SERial:FLEXray Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:FLEXray:CYCLE <cycle> (see page 967)	:SEARch:SERial:FLEXray:CYCLE? (see page 967)	<cycle> ::= {ALL   <cycle #>} <cycle #> ::= integer from 0-63
:SEARch:SERial:FLEXray:DATA <string> (see page 968)	:SEARch:SERial:FLEXray:DATA? (see page 968)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X }
:SEARch:SERial:FLEXray:DATA:LENGTH <length> (see page 969)	:SEARch:SERial:FLEXray:DATA:LENGTH? (see page 969)	<length> ::= integer from 1 to 12 in NR1 format

**Table 45** :SEARch:SERial:FLEXray Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARch:SERial:FLEXray:FRAMe <frame id> (see page 970)	:SEARch:SERial:FLEXray:FRAMe? (see page 970)	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047
:SEARch:SERial:FLEXray:MODE <value> (see page 971)	:SEARch:SERial:FLEXray:MODE? (see page 971)	<value> ::= {FRAMe   CYCLE   DATA   HERRor   FERRor   AERRor}

**Table 46** :SEARch:SERial:I2S Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:I2S:AUDIO <audio_ch> (see page 973)	:SEARch:SERial:I2S:AUDIO? (see page 973)	<audio_ch> ::= {RIGHT   LEFT   EITHer}
:SEARch:SERial:I2S:MODE <value> (see page 974)	:SEARch:SERial:I2S:MODE? (see page 974)	<value> ::= {EQUAL   NOTEQUAL   LESSthan   GREATERthan   INRange   OUTRange}
:SEARch:SERial:I2S:ATTern:DATA <string> (see page 975)	:SEARch:SERial:I2S:ATTern:DATA? (see page 975)	<string> ::= "n" where n ::= 32-bit integer in signed decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   X} when <base> = BINARY <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} when <base> = HEX
:SEARch:SERial:I2S:ATTern:FORMAT <base> (see page 976)	:SEARch:SERial:I2S:ATTern:FORMAT? (see page 976)	<base> ::= {BINARY   HEX   DECIMAL}
:SEARch:SERial:I2S:RANGE <lower>, <upper> (see page 977)	:SEARch:SERial:I2S:RANGE? (see page 977)	<lower> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <upper> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal

**Table 47** :SEARch:SERial:IIC Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:IIC:MO DE <value> (see <a href="#">page 979</a> )	:SEARch:SERial:IIC:MO DE? (see <a href="#">page 979</a> )	<value> ::= { READ7   WRITE7   NACKnowledge   ANACK   R7Data2   W7Data2   RESTart   READEprom}
:SEARch:SERial:IIC:PA TTern:ADDReSS <value> (see <a href="#">page 981</a> )	:SEARch:SERial:IIC:PA TTern:ADDReSS? (see <a href="#">page 981</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:PA TTern:DATA <value> (see <a href="#">page 982</a> )	:SEARch:SERial:IIC:PA TTern:DATA? (see <a href="#">page 982</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:PA TTern:DATA2 <value> (see <a href="#">page 983</a> )	:SEARch:SERial:IIC:PA TTern:DATA2? (see <a href="#">page 983</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:QU ALifier <value> (see <a href="#">page 984</a> )	:SEARch:SERial:IIC:QU ALifier? (see <a href="#">page 984</a> )	<value> ::= { EQUAL   NOTEQUAL   LESSthan   GREATERthan}

**Table 48** :SEARch:SERial:LIN Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:LIN:ID <value> (see <a href="#">page 986</a> )	:SEARch:SERial:LIN:ID ? (see <a href="#">page 986</a> )	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f (with Option AMS) <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9   A,...,F} for hexadecimal
:SEARch:SERial:LIN:MO DE <value> (see <a href="#">page 987</a> )	:SEARch:SERial:LIN:MO DE? (see <a href="#">page 987</a> )	<value> ::= { ID   DATA   ERROR}

**Table 48** :SEARch:SERial:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARch:SERial:LIN:PA TTern:DATA <string> (see <a href="#">page 988</a> )	:SEARch:SERial:LIN:PA TTern:DATA? (see <a href="#">page 988</a> )	When :SEARch:SERial:LIN:PATTERn:FORMat DECimal, <string> ::= "n" where n ::= 32-bit integer in unsigned decimal, returns "\$" if data has any don't cares When :SEARch:SERial:LIN:PATTERn:FORMat HEX, <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X}
:SEARch:SERial:LIN:PA TTern:DATA:LENGTH <length> (see <a href="#">page 989</a> )	:SEARch:SERial:LIN:PA TTern:DATA:LENGTH? (see <a href="#">page 989</a> )	<length> ::= integer from 1 to 8 in NR1 format
:SEARch:SERial:LIN:PA TTern:FORMAT <base> (see <a href="#">page 990</a> )	:SEARch:SERial:LIN:PA TTern:FORMAT? (see <a href="#">page 990</a> )	<base> ::= {HEX   DECimal}
:SEARch:SERial:LIN:SY MBolic:FRAMe <name> (see <a href="#">page 991</a> )	:SEARch:SERial:LIN:SY MBolic:FRAMe? (see <a href="#">page 991</a> )	<name> ::= quoted ASCII string
:SEARch:SERial:LIN:SY MBolic:SIGNal <name> (see <a href="#">page 992</a> )	:SEARch:SERial:LIN:SY MBolic:SIGNal? (see <a href="#">page 992</a> )	<name> ::= quoted ASCII string
:SEARch:SERial:LIN:SY MBolic:VALue <data> (see <a href="#">page 993</a> )	:SEARch:SERial:LIN:SY MBolic:VALue? (see <a href="#">page 993</a> )	<data> ::= value in NR3 format

**Table 49** :SEARch:SERial:M1553 Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:M1553: MODE <value> (see <a href="#">page 995</a> )	:SEARch:SERial:M1553: MODE? (see <a href="#">page 995</a> )	<value> ::= {DSTArt   CSTArt   RTA   RTA11   PERRor   SERRor   MERRor}

**Table 49** :SEARch:SERial:M1553 Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARch:SERial:M1553:PATTern:DATA <string> (see page 996)	:SEARch:SERial:M1553:PATTern:DATA? (see page 996)	<string> ::= "nn...n" where n ::= {0   1}
:SEARch:SERial:M1553:RTA <value> (see page 997)	:SEARch:SERial:M1553:RTA? (see page 997)	<value> ::= 5-bit integer in decimal, <hexadecimal>, <binary>, or <string> from 0-31 <hexadecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} <binary> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9 A,...,F}

**Table 50** :SEARch:SERial:SENT Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:SENT:FAST:DATA <string> (see page 999)	:SEARch:SERial:SENT:FAST:DATA? (see page 999)	<string> ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}
:SEARch:SERial:SENT:MODE <mode> (see page 1000)	:SEARch:SERial:SENT:MODE? (see page 1000)	<mode> ::= {FCData   SCMid   SCData   CRCerror PPERror}
:SEARch:SERial:SENT:SLOW:DATA <data> (see page 1001)	:SEARch:SERial:SENT:SLOW:DATA? (see page 1001)	<data> ::= from -1 (don't care) to 65535, in NR1 format.
:SEARch:SERial:SENT:SLOW:ID <id> (see page 1002)	:SEARch:SERial:SENT:SLOW:ID? (see page 1002)	<id> ::= from -1 (don't care) to 255, in NR1 format.

**Table 51** :SEARch:SERial:SPI Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:SPI:MODE <value> (see page 1004)	:SEARch:SERial:SPI:MODE? (see page 1004)	<value> ::= {MOSI   MISO}
:SEARch:SERial:SPI:PTTern:DATA <string> (see page 1005)	:SEARch:SERial:SPI:PTTern:DATA? (see page 1005)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X}
:SEARch:SERial:SPI:PTTern:WIDTH <width> (see page 1006)	:SEARch:SERial:SPI:PTTern:WIDTH? (see page 1006)	<width> ::= integer from 1 to 10

**Table 52** :SEARch:SERial:UART Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:UART:D ATA <value> (see <a href="#">page 1008</a> )	:SEARch:SERial:UART:D ATA? (see <a href="#">page 1008</a> )	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, <binary>, or <quoted_string> format <hexadecimal> ::= #Hnn where n ::= {0,...,9  A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0   1} for binary <quoted_string> ::= any of the 128 valid 7-bit ASCII characters (or standard abbreviations)
:SEARch:SERial:UART:M ODE <value> (see <a href="#">page 1009</a> )	:SEARch:SERial:UART:M ODE? (see <a href="#">page 1009</a> )	<value> ::= {RDATA   RD1   RD0   RDX   TDATA   TD1   TD0   TDX   PARityerror   AERRor}
:SEARch:SERial:UART:Q UALifier <value> (see <a href="#">page 1010</a> )	:SEARch:SERial:UART:Q UALifier? (see <a href="#">page 1010</a> )	<value> ::= {EQUAL   NOTEqual   GREaterthan   LESSthan}

**Table 53** :SYSTem Commands Summary

Command	Query	Options and Query Returns
:SYSTem:DATE <date> (see <a href="#">page 1013</a> )	:SYSTem:DATE? (see <a href="#">page 1013</a> )	<date> ::= <year>,<month>,<day> <year> ::= 4-digit year in NR1 format <month> ::= {1,...,12   JANuary   FEBruary   MARch   APRil   MAY   JUNe   JULy   AUGust   SEPtember   OCTober   NOVember   DECember} <day> ::= {1,...31}
:SYSTem:DSP <string> (see <a href="#">page 1014</a> )	n/a	<string> ::= up to 75 characters as a quoted ASCII string
n/a	:SYSTem:ERRor? (see <a href="#">page 1015</a> )	<error> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see <a href="#">page 1291</a> ).
:SYSTem:LOCK <value> (see <a href="#">page 1016</a> )	:SYSTem:LOCK? (see <a href="#">page 1016</a> )	<value> ::= {{1   ON}   {0   OFF}}

**Table 53** :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:PERSONa [:MANufacturer] <manufacturer_string> (see <a href="#">page 1017</a> )	:SYSTem:PERSONa [:MANufacturer]? (see <a href="#">page 1017</a> )	<manufacturer_string> ::= quoted ASCII string, up to 63 characters
:SYSTem:PERSONa [:MANufacturer] :DEFault (see <a href="#">page 1018</a> )	n/a	Sets manufacturer string to "KEYSIGHT TECHNOLOGIES"
:SYSTem:PRESet (see <a href="#">page 1019</a> )	n/a	See :SYSTem:PRESet (see <a href="#">page 1019</a> )
:SYSTem:PROTection:LOCK <value> (see <a href="#">page 1022</a> )	:SYSTem:PROTection:LOCK? (see <a href="#">page 1022</a> )	<value> ::= {{1   ON}   {0   OFF}}
:SYSTem:RLOGger <setting>[,<file_name>[,<write_mode>]] (see <a href="#">page 1023</a> )	n/a	<setting> ::= {{0   OFF}   {1   ON}} <file_name> ::= quoted ASCII string <write_mode> ::= {CREATE   APPEND}
:SYSTem:RLOGger:DESTination <dest> (see <a href="#">page 1024</a> )	:SYSTem:RLOGger:DESTination? (see <a href="#">page 1024</a> )	<dest> ::= {FILE   SCReen   BOTH}
:SYSTem:RLOGger:DISPLAY {{0   OFF}   {1   ON}} (see <a href="#">page 1025</a> )	:SYSTem:RLOGger:DISPLAY? (see <a href="#">page 1025</a> )	<setting> ::= {0   1}
:SYSTem:RLOGger:FNAMe <file_name> (see <a href="#">page 1026</a> )	:SYSTem:RLOGger:FNAMe? (see <a href="#">page 1026</a> )	<file_name> ::= quoted ASCII string
:SYSTem:RLOGger:STATE {{0   OFF}   {1   ON}} (see <a href="#">page 1027</a> )	:SYSTem:RLOGger:STATE? (see <a href="#">page 1027</a> )	<setting> ::= {0   1}
:SYSTem:RLOGger:TRANsport {{0   OFF}   {1   ON}} (see <a href="#">page 1028</a> )	:SYSTem:RLOGger:TRANsport? (see <a href="#">page 1028</a> )	<setting> ::= {0   1}
:SYSTem:RLOGger:WMODe <write_mode> (see <a href="#">page 1029</a> )	:SYSTem:RLOGger:WMODe? (see <a href="#">page 1029</a> )	<write_mode> ::= {CREATE   APPEND}
:SYSTem:SETup <setup_data> (see <a href="#">page 1030</a> )	:SYSTem:SETup? (see <a href="#">page 1030</a> )	<setup_data> ::= data in IEEE 488.2 # format.

**Table 53** :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:TIME <time> (see <a href="#">page 1032</a> )	:SYSTem:TIME? (see <a href="#">page 1032</a> )	<time> ::= hours,minutes,seconds in NR1 format
:SYSTem:TOUCH {{1   ON}   {0   OFF}} (see <a href="#">page 1033</a> )	:SYSTem:TOUCH? (see <a href="#">page 1033</a> )	{1   0}

**Table 54** :TIMEbase Commands Summary

Command	Query	Options and Query Returns
:TIMEbase:MODE <value> (see <a href="#">page 1037</a> )	:TIMEbase:MODE? (see <a href="#">page 1037</a> )	<value> ::= {MAIN   WINDOW   XY   ROLL}
:TIMEbase:POSIon <pos> (see <a href="#">page 1038</a> )	:TIMEbase:POSIon? (see <a href="#">page 1038</a> )	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMEbase:RANGE <range_value> (see <a href="#">page 1039</a> )	:TIMEbase:RANGE? (see <a href="#">page 1039</a> )	<range_value> ::= time for 10 div in seconds in NR3 format
:TIMEbase:REFERENCE {LEFT   CENTER   RIGHT} (see <a href="#">page 1040</a> )	:TIMEbase:REFERENCE? (see <a href="#">page 1040</a> )	<return_value> ::= {LEFT   CENTER   RIGHT}
:TIMEbase:SCALe <scale_value> (see <a href="#">page 1041</a> )	:TIMEbase:SCALe? (see <a href="#">page 1041</a> )	<scale_value> ::= time/div in seconds in NR3 format
:TIMEbase:VERNier {{0   OFF}   {1   ON}} (see <a href="#">page 1042</a> )	:TIMEbase:VERNier? (see <a href="#">page 1042</a> )	{0   1}
:TIMEbase:WINDOW:POSIon <pos> (see <a href="#">page 1043</a> )	:TIMEbase:WINDOW:POSIon? (see <a href="#">page 1043</a> )	<pos> ::= time from the trigger event to the zoomed view reference point in NR3 format
:TIMEbase:WINDOW:RANGE <range_value> (see <a href="#">page 1044</a> )	:TIMEbase:WINDOW:RANGE? (see <a href="#">page 1044</a> )	<range value> ::= range value in seconds in NR3 format for the zoomed window
:TIMEbase:WINDOW:SCALe <scale_value> (see <a href="#">page 1045</a> )	:TIMEbase:WINDOW:SCALe? (see <a href="#">page 1045</a> )	<scale_value> ::= scale value in seconds in NR3 format for the zoomed window

**Table 55** General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:FORCe (see page 1050)	n/a	n/a
:TRIGger:HFReject {{0   OFF}   {1   ON}} (see page 1051)	:TRIGger:HFReject? (see page 1051)	{0   1}
:TRIGger:HOLDoff <holdoff_time> (see page 1052)	:TRIGger:HOLDoff? (see page 1052)	<holdoff_time> ::= 60 ns to 10 s in NR3 format
:TRIGger:LEVel:ASETup (see page 1053)	n/a	n/a
:TRIGger:LEVel:HIGH <level>, <source> (see page 1054)	:TRIGger:LEVel:HIGH? <source> (see page 1054)	<level> ::= .75 x full-scale voltage from center screen in NR3 format. <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:LEVel:LOW <level>, <source> (see page 1055)	:TRIGger:LEVel:LOW? <source> (see page 1055)	<level> ::= .75 x full-scale voltage from center screen in NR3 format. <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:MODE <mode> (see page 1056)	:TRIGger:MODE? (see page 1056)	<mode> ::= {EDGE   GLITch   PATTern   TV   DELay   EBURst   OR   RUNT   SHOld   TRANSition   SBUS{1   2}} <return_value> ::= {<mode>   <none>} <none> ::= query returns "NONE" if the :TIMEbase:MODE is ROLL or XY
:TRIGger:NREJect {{0   OFF}   {1   ON}} (see page 1057)	:TRIGger:NREJect? (see page 1057)	{0   1}
:TRIGger:SWEep <sweep> (see page 1058)	:TRIGger:SWEep? (see page 1058)	<sweep> ::= {AUTO   NORMAL}

**Table 56** :TRIGger:DELay Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DELay:ARM:SL OPe <slope> (see <a href="#">page 1060</a> )	:TRIGger:DELay:ARM:SL OPe? (see <a href="#">page 1060</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:DELay:ARM:SO URce <source> (see <a href="#">page 1061</a> )	:TRIGger:DELay:ARM:SO URce? (see <a href="#">page 1061</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:DELay:TDElay :TIME <time_value> (see <a href="#">page 1062</a> )	:TRIGger:DELay:TDElay :TIME? (see <a href="#">page 1062</a> )	<time_value> ::= time in seconds in NR3 format
:TRIGger:DELay:TRIGge r:COUNT <count> (see <a href="#">page 1063</a> )	:TRIGger:DELay:TRIGge r:COUNT? (see <a href="#">page 1063</a> )	<count> ::= integer in NR1 format
:TRIGger:DELay:TRIGge r:SLOPe <slope> (see <a href="#">page 1064</a> )	:TRIGger:DELay:TRIGge r:SLOPe? (see <a href="#">page 1064</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:DELay:TRIGge r:SOURce <source> (see <a href="#">page 1065</a> )	:TRIGger:DELay:TRIGge r:SOURce? (see <a href="#">page 1065</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 57** :TRIGger:EBURst Commands Summary

Command	Query	Options and Query Returns
:TRIGger:EBURst:COUNT <count> (see <a href="#">page 1067</a> )	:TRIGger:EBURst:COUNT ? (see <a href="#">page 1067</a> )	<count> ::= integer in NR1 format
:TRIGger:EBURst:IDLE <time_value> (see <a href="#">page 1068</a> )	:TRIGger:EBURst:IDLE? (see <a href="#">page 1068</a> )	<time_value> ::= time in seconds in NR3 format

**Table 57** :TRIGger:EBURst Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:EBURst:SLOPe <slope> (see page 1069)	:TRIGger:EBURst:SLOPe ? (see <a href="#">page 1069</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:EBURst:SOURce <source> (see page 1070)	:TRIGger:EBURst:SOURce? (see <a href="#">page 1070</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 58** :TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:COUPLing {AC   DC   LFReject} (see <a href="#">page 1072</a> )	:TRIGger[:EDGE]:COUPLing? (see <a href="#">page 1072</a> )	{AC   DC   LFReject}
:TRIGger[:EDGE]:LEVel <level> [,<source>] (see <a href="#">page 1073</a> )	:TRIGger[:EDGE]:LEVel? [<source>] (see <a href="#">page 1073</a> )	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers, <level> ::= ±(external range setting) in NR3 format. For digital channels (MSO models), <level> ::= ±8 V. <source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   EXTernal } for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger[:EDGE]:REJect {OFF   LFReject   HFReject} (see <a href="#">page 1074</a> )	:TRIGger[:EDGE]:REJect? (see <a href="#">page 1074</a> )	{OFF   LFReject   HFReject}

**Table 58** :TRIGger[:EDGE] Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:SLOPe <polarity> (see page 1075)	:TRIGger[:EDGE]:SLOPe? ? (see <a href="#">page 1075</a> )	<polarity> ::= {POSitive   NEGative   EITHer   ALTernate}
:TRIGger[:EDGE]:SOURce <source> (see page 1076)	:TRIGger[:EDGE]:SOURce? ? (see <a href="#">page 1076</a> )	<source> ::= {CHANnel<n>   EXTERNAL   LINE   WGEN} for the DSO models  <source> ::= {CHANnel<n>   DIGItal<d>   EXTERNAL   LINE   WGEN} for the MSO models  <n> ::= 1 to (# analog channels) in NR1 format  <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 59** :TRIGger:GLITch Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITch:GREaterthan <greater_than_time>[suffix] (see <a href="#">page 1079</a> )	:TRIGger:GLITch:GREaterthan? ? (see <a href="#">page 1079</a> )	<greater_than_time> ::= floating-point number in NR3 format  [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:LESSthan <less_than_time>[suffix] (see <a href="#">page 1080</a> )	:TRIGger:GLITch:LESSthan? ? (see <a href="#">page 1080</a> )	<less_than_time> ::= floating-point number in NR3 format  [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:LEVel <level> [<source>] (see <a href="#">page 1081</a> )	:TRIGger:GLITch:LEVel? ? (see <a href="#">page 1081</a> )	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format.  For external triggers (DSO models), <level> ::= ±(external range setting) in NR3 format.  For digital channels (MSO models), <level> ::= ±8 V.  <source> ::= {CHANnel<n>   EXTERNAL} for DSO models  <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models  <n> ::= 1 to (# analog channels) in NR1 format  <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 59** :TRIGger:GLITch Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:GLITch:POLarity <polarity> (see page 1082)	:TRIGger:GLITch:POLarity? (see page 1082)	<polarity> ::= {POSitive   NEGative}
:TRIGger:GLITch:QUALifier <qualifier> (see page 1083)	:TRIGger:GLITch:QUALifier? (see page 1083)	<qualifier> ::= {GREaterthan   LESSthan   RANGE}
:TRIGger:GLITch:RANGE <less_than_time>[suffix], <greater_than_time>[suffix] (see page 1084)	:TRIGger:GLITch:RANGE? (see page 1084)	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:SOURce <source> (see page 1085)	:TRIGger:GLITch:SOURce? (see page 1085)	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 60** :TRIGger:NFC Commands Summary

Command	Query	Options and Query Returns
:TRIGger:NFC:AEVENT <arm_event> (see page 1087)	:TRIGger:NFC:AEVENT? (see page 1087)	<arm_event> ::= {NONE   ASReq   AALLreq   AEITher   BSReq   BALLreq   BEITher   FSReq}
n/a	:TRIGger:NFC:ATTIME? (see page 1088)	<time> ::= seconds in NR3 format
:TRIGger:NFC:SOURCE <source> (see page 1089)	:TRIGger:NFC:SOURce? (see page 1089)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:NFC:STANDARD <standard> (see page 1090)	:TRIGger:NFC:STANDARD? (see page 1090)	<standard> ::= {{A   A106}   {B   B106}   F212   F424}
:TRIGger:NFC:TEVENT <trigger_event> (see page 1091)	:TRIGger:NFC:TEVENT? (see page 1091)	<trigger_event> ::= {ATRigger   ASReq   AALLreq   AEITher   ASDDreq   BSReq   BALLreq   BEITher   BATTrib   FSReq   FAReq   FPRreamble}
n/a	:TRIGger:NFC:TIMeout? (see page 1093)	{0   1}

**Table 60** :TRIGger:NFC Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:NFC:TIMEout:ENABLE {{0   OFF}   {1   ON}} (see page 1094)	:TRIGger:NFC:TIMEout:ENABLE? (see page 1094)	{0   1}
:TRIGger:NFC:TIMEout:TIME <time> (see page 1095)	:TRIGger:NFC:TIMEout:TIME? (see page 1095)	<time> ::= seconds in NR3 format

**Table 61** :TRIGger:OR Commands Summary

Command	Query	Options and Query Returns
:TRIGger:OR <string> (see page 1097)	:TRIGger:OR? (see page 1097)	<p>&lt;string&gt; ::= "nn...n" where n ::= {R   F   E   X}</p> <p>R = rising edge, F = falling edge, E = either edge, X = don't care.</p> <p>Each character in the string is for an analog or digital channel as shown on the front panel display.</p>

**Table 62** :TRIGger:PATTERn Commands Summary

Command	Query	Options and Query Returns
:TRIGger:PATTERn <string>[,<edge_source>,<edge>] (see page 1099)	:TRIGger:PATTERn? (see page 1100)	<p>&lt;string&gt; ::= "nn...n" where n ::= {0   1   X   R   F} when &lt;base&gt; = ASCII &lt;string&gt; ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$} when &lt;base&gt; = HEX</p> <p>&lt;edge_source&gt; ::= {CHANnel&lt;n&gt;   NONE} for DSO models</p> <p>&lt;edge_source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   NONE} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;edge&gt; ::= {POSiTive   NEGative}</p>
:TRIGger:PATTERn:FORM at <base> (see page 1101)	:TRIGger:PATTERn:FORM at? (see page 1101)	<base> ::= {ASCII   HEX}

**Table 62** :TRIGger:PATTERn Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:PATTERn:GREaterthan <greater_than_time>[suffix] (see <a href="#">page 1102</a> )	:TRIGger:PATTERn:GREaterthan? (see <a href="#">page 1102</a> )	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:PATTERn:LESSthan <less_than_time>[suffix] (see <a href="#">page 1103</a> )	:TRIGger:PATTERn:LESSthan? (see <a href="#">page 1103</a> )	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:PATTERn:QUALifier <qualifier> (see <a href="#">page 1104</a> )	:TRIGger:PATTERn:QUALifier? (see <a href="#">page 1104</a> )	<qualifier> ::= {ENTERed   GREaterthan   LESSthan   INRange   OUTRange   TIMEout}
:TRIGger:PATTERn:RANGE <less_than_time>[suffix], <greater_than_time>[suffix] (see <a href="#">page 1105</a> )	:TRIGger:PATTERn:RANGE? (see <a href="#">page 1105</a> )	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 63** :TRIGger:RUNT Commands Summary

Command	Query	Options and Query Returns
:TRIGger:RUNT:POLarity <polarity> (see <a href="#">page 1107</a> )	:TRIGger:RUNT:POLarity? (see <a href="#">page 1107</a> )	<polarity> ::= {POSitive   NEGative   EITHer}
:TRIGger:RUNT:QUALifier <qualifier> (see <a href="#">page 1108</a> )	:TRIGger:RUNT:QUALifier? (see <a href="#">page 1108</a> )	<qualifier> ::= {GREaterthan   LESSthan   NONE}
:TRIGger:RUNT:SOURce <source> (see <a href="#">page 1109</a> )	:TRIGger:RUNT:SOURce? (see <a href="#">page 1109</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:RUNT:TIME <time>[suffix] (see <a href="#">page 1110</a> )	:TRIGger:RUNT:TIME? (see <a href="#">page 1110</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 64** :TRIGger:SHOLD Commands Summary

Command	Query	Options and Query Returns
:TRIGger:SHOLD:SLOPe <slope> (see page 1112)	:TRIGger:SHOLD:SLOPe? (see page 1112)	<slope> ::= {NEGative   POSitive}
:TRIGger:SHOLD:SOURce :CLOCK <source> (see page 1113)	:TRIGger:SHOLD:SOURce :CLOCK? (see page 1113)	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:SHOLD:SOURce :DATA <source> (see page 1114)	:TRIGger:SHOLD:SOURce :DATA? (see page 1114)	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:SHOLD:TIME:H OLD <time>[suffix] (see page 1115)	:TRIGger:SHOLD:TIME:H OLD? (see page 1115)	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:SHOLD:TIME:S ETup <time>[suffix] (see page 1116)	:TRIGger:SHOLD:TIME:S ETup? (see page 1116)	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 65** :TRIGger:TRANSition Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TRANSition:Q UALifier <qualifier> (see page 1118)	:TRIGger:TRANSition:Q UALifier? (see page 1118)	<qualifier> ::= {GREaterthan   LESSthan}
:TRIGger:TRANSition:S LOPe <slope> (see page 1119)	:TRIGger:TRANSition:S LOPe? (see page 1119)	<slope> ::= {NEGative   POSitive}
:TRIGger:TRANSition:S OURce <source> (see page 1120)	:TRIGger:TRANSition:S OURce? (see page 1120)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TRANSition:T IME <time>[suffix] (see page 1121)	:TRIGger:TRANSition:T IME? (see page 1121)	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**Table 66** :TRIGger:TV Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TV:LINE <line number> (see <a href="#">page 1123</a> )	:TRIGger:TV:LINE? (see <a href="#">page 1123</a> )	<line number> ::= integer in NR1 format
:TRIGger:TV:MODE <tv mode> (see <a href="#">page 1124</a> )	:TRIGger:TV:MODE? (see <a href="#">page 1124</a> )	<tv mode> ::= {FIELD1   FIELD2   AFIELDS   ALINES   LINE   LFIELD1   LFIELD2   LATERNATE}
:TRIGger:TV:POLarity <polarity> (see <a href="#">page 1125</a> )	:TRIGger:TV:POLarity? (see <a href="#">page 1125</a> )	<polarity> ::= {POSITIVE   NEGATIVE}
:TRIGger:TV:SOURce <source> (see <a href="#">page 1126</a> )	:TRIGger:TV:SOURce? (see <a href="#">page 1126</a> )	<source> ::= {CHANNEL<n>} <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TV:STANDARD <standard> (see <a href="#">page 1127</a> )	:TRIGger:TV:STANDARD? (see <a href="#">page 1127</a> )	<standard> ::= {NTSC   PAL   PALM   SECAM} <standard> ::= {GENERIC   {P480L60HZ   P480}   {P720L60HZ   P720}   {P1080L24HZ   P1080}   P1080L25HZ   P1080L50HZ   P1080L60HZ   {I1080L50HZ   I1080}   I1080L60HZ} with extended video triggering license
:TRIGger:TV:UDTV:ENUM ber <count> (see <a href="#">page 1128</a> )	:TRIGger:TV:UDTV:ENUM ber? (see <a href="#">page 1128</a> )	<count> ::= edge number in NR1 format
:TRIGger:TV:UDTV:HSYN C {{0   OFF}   {1   ON}} (see <a href="#">page 1129</a> )	:TRIGger:TV:UDTV:HSYN C? (see <a href="#">page 1129</a> )	{0   1}
:TRIGger:TV:UDTV:HTIM e <time> (see <a href="#">page 1130</a> )	:TRIGger:TV:UDTV:HTIM e? (see <a href="#">page 1130</a> )	<time> ::= seconds in NR3 format
:TRIGger:TV:UDTV:PGTH an <min_time> (see <a href="#">page 1131</a> )	:TRIGger:TV:UDTV:PGTH an? (see <a href="#">page 1131</a> )	<min_time> ::= seconds in NR3 format

**Table 67** :TRIGger:USB Commands Summary

Command	Query	Options and Query Returns
:TRIGger:USB:SOURce:D MINus <source> (see <a href="#">page 1133</a> )	:TRIGger:USB:SOURce:D MINus? (see <a href="#">page 1133</a> )	<source> ::= {CHANnel<n>   EXTernal} for the DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:USB:SOURce:D PLus <source> (see <a href="#">page 1134</a> )	:TRIGger:USB:SOURce:D PLus? (see <a href="#">page 1134</a> )	<source> ::= {CHANnel<n>   EXTernal} for the DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:USB:SPEEd <value> (see <a href="#">page 1135</a> )	:TRIGger:USB:SPEEd? (see <a href="#">page 1135</a> )	<value> ::= {LOW   FULL}
:TRIGger:USB:TRIGger <value> (see <a href="#">page 1136</a> )	:TRIGger:USB:TRIGger? (see <a href="#">page 1136</a> )	<value> ::= {SOP   EOP   ENTerSuspend   EXITsuspend   RESet}

**Table 68** :TRIGger:ZONE Commands Summary

Command	Query	Options and Query Returns
:TRIGger:ZONE:SOURce <source> (see <a href="#">page 1138</a> )	:TRIGger:ZONE:SOURce? (see <a href="#">page 1138</a> )	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:ZONE:STATe { {0   OFF}   {1   ON} } (see <a href="#">page 1139</a> )	:TRIGger:ZONE:STATe? (see <a href="#">page 1139</a> )	{0   1}
:TRIGger:ZONE<n>:MODE <mode> (see <a href="#">page 1140</a> )	:TRIGger:ZONE<n>:MODE ? (see <a href="#">page 1140</a> )	<mode> ::= {INTersect   NOTintersect} <n> ::= 1-2 in NR1 format

**Table 68** :TRIGger:ZONE Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:ZONE<n>:PLACEMENT <width>, <height>, <x_center>, <y_center> (see page 1141)	:TRIGger:ZONE<n>:PLACEMENT? (see page 1141)	<width> ::= width of zone in seconds <height> ::= height of zone in volts <x_center> ::= center of zone in seconds <y_center> ::= center of zone in volts <n> ::= 1-2 in NR1 format
n/a	:TRIGger:ZONE<n>:VALIDITY? (see page 1142)	<value> ::= {VALid   INValid   OSCreen} <n> ::= 1-2 in NR1 format
:TRIGger:ZONE<n>:STATE {{0   OFF}   {1   ON}} (see page 1143)	:TRIGger:ZONE<n>:STATE? (see page 1143)	{0   1} <n> ::= 1-2 in NR1 format

**Table 69** :WAVEform Commands Summary

Command	Query	Options and Query Returns
:WAVEform:BYTeorder <value> (see page 1153)	:WAVEform:BYTeorder? (see page 1153)	<value> ::= {LSBFFirst   MSBFFirst}
n/a	:WAVEform:COUNT? (see page 1154)	<count> ::= an integer from 1 to 65536 in NR1 format
n/a	:WAVEform:DATA? (see page 1155)	<binary block length bytes>, <binary data> For example, to transmit 1000 bytes of data, the syntax would be: #800001000<1000 bytes of data><NL> 8 is the number of digits that follow 00001000 is the number of bytes to be transmitted <1000 bytes of data> is the actual data
:WAVEform:FORMAT <value> (see page 1157)	:WAVEform:FORMAT? (see page 1157)	<value> ::= {WORD   BYTE   ASCII}

**Table 69** :WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVEform:POINTs <# points> (see <a href="#">page 1158</a> )	:WAVEform:POINTs? (see <a href="#">page 1158</a> )	<# points> ::= {100   250   500   1000   <points_mode>} if waveform points mode is NORMAl <# points> ::= {100   250   500   1000   2000 ... 8000000 in 1-2-5 sequence   <points_mode>} if waveform points mode is MAXimum or RAW <points_mode> ::= {NORMAl   MAXimum   RAW}
:WAVEform:POINTs:MODE <points_mode> (see <a href="#">page 1160</a> )	:WAVEform:POINTs:MODE? (see <a href="#">page 1160</a> )	<points_mode> ::= {NORMAl   MAXimum   RAW}
n/a	:WAVEform:PREamble? (see <a href="#">page 1162</a> )	<preamble_block> ::= <format NR1>, <type NR1>, <points NR1>, <count NR1>, <xincrement NR3>, <xorigin NR3>, <xreference NR1>, <yincrement NR3>, <yorigin NR3>, <yreference NR1> <format> ::= an integer in NR1 format: <ul style="list-style-type: none"><li>• 0 for BYTE format</li><li>• 1 for WORD format</li><li>• 2 for ASCii format</li></ul> <type> ::= an integer in NR1 format: <ul style="list-style-type: none"><li>• 0 for NORMAL type</li><li>• 1 for PEAK detect type</li><li>• 3 for AVERAGE type</li><li>• 4 for HRESolution type</li></ul> <count> ::= Average count, or 1 if PEAK detect type or NORMAL; an integer in NR1 format
n/a	:WAVEform:SEGmented:COUNT? (see <a href="#">page 1165</a> )	<count> ::= an integer from 2 to 1000 in NR1 format (with Option SGM)
n/a	:WAVEform:SEGmented:TAG? (see <a href="#">page 1166</a> )	<time_tag> ::= in NR3 format (with Option SGM)

**Table 69** :WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVEform:SOURce <source> (see page 1167)	:WAVEform:SOURce? (see <a href="#">page 1167</a> )	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:WAVEform:SOURce:SUBS ource <subsource> (see <a href="#">page 1171</a> )	:WAVEform:SOURce:SUBS ource? (see <a href="#">page 1171</a> )	<subsource> ::= {{SUB0   RX   MOSI}   {SUB1   TX   MISO}}
n/a	:WAVEform:TYPE? (see <a href="#">page 1172</a> )	<return_mode> ::= {NORM   PEAK   AVER   HRES}
:WAVEform:UNSIGNED { {0   OFF}   {1   ON} } (see <a href="#">page 1173</a> )	:WAVEform:UNSIGNED? (see <a href="#">page 1173</a> )	{0   1}
:WAVEform:VIEW <view> (see <a href="#">page 1174</a> )	:WAVEform:VIEW? (see <a href="#">page 1174</a> )	<view> ::= {MAIN}
n/a	:WAVEform:XINCREMENT? (see <a href="#">page 1175</a> )	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVEform:XORIGIN? (see <a href="#">page 1176</a> )	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVEform:XREFERENCE? (see <a href="#">page 1177</a> )	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)
n/a	:WAVEform:YINCREMENT? (see <a href="#">page 1178</a> )	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVEform:YORIGIN? (see <a href="#">page 1179</a> )	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVEform:YREFERENCE? (see <a href="#">page 1180</a> )	<return_value> ::= y-reference value in the current preamble in NR1 format

**Table 70** :WGEN<w> Commands Summary

Command	Query	Options and Query Returns
:WGEN<w>:ARBitrary:BY Teorder <order> (see <a href="#">page 1185</a> )	:WGEN<w>:ARBitrary:BY Teorder? (see <a href="#">page 1185</a> )	<order> ::= {MSBFIRST   LSBFIRST} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DA TA {<binary>   <value>, <value> ... } (see <a href="#">page 1186</a> )	n/a	<binary> ::= floating point values between -1.0 to +1.0 in IEEE 488.2 binary block format <value> ::= floating point values between -1.0 to +1.0 in comma-separated format <w> ::= 1 or 2 in NR1 format
n/a	:WGEN<w>:ARBitrary:DA TA:ATTRibute:POINts? (see <a href="#">page 1187</a> )	<points> ::= number of points in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DA TA:CLEAR (see <a href="#">page 1188</a> )	n/a	<w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DA TA:DAC {<binary>   <value>, <value> ... } (see <a href="#">page 1189</a> )	n/a	<binary> ::= decimal 16-bit integer values between -512 to +511 in IEEE 488.2 binary block format <value> ::= decimal integer values between -512 to +511 in comma-separated NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:IN Terpolate {{0   OFF}   {1   ON}} (see <a href="#">page 1190</a> )	:WGEN<w>:ARBitrary:IN Terpolate? (see <a href="#">page 1190</a> )	{0   1} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:ST ORe <source> (see <a href="#">page 1191</a> )	n/a	<source> ::= {CHANnel<n>   WMEMory<r>   FUNCtion<m>   FFT   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FREQuency <frequency> (see <a href="#">page 1192</a> )	:WGEN<w>:FREQuency? (see <a href="#">page 1192</a> )	<frequency> ::= frequency in Hz in NR3 format <w> ::= 1 or 2 in NR1 format

**Table 70** :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:FUNCTION <signal> (see page 1193)	:WGEN<w>:FUNCTION? (see <a href="#">page 1196</a> )	<signal> ::= {SINusoid   SQuare   RAMP   PULSe   NOISe   DC   SINC   EXPrise   EXPFall   CARDiac   GAUSSian   ARBItrary} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTION:PULSE:WIDTH <width> (see page 1197)	:WGEN<w>:FUNCTION:PULSE:WIDTH? (see <a href="#">page 1197</a> )	<width> ::= pulse width in seconds in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTION:RAMP:SYMMetry <percent> (see <a href="#">page 1198</a> )	:WGEN<w>:FUNCTION:RAMP:SYMMetry? (see <a href="#">page 1198</a> )	<percent> ::= symmetry percentage from 0% to 100% in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTION:SQUARE:DCYCLE <percent> (see <a href="#">page 1199</a> )	:WGEN<w>:FUNCTION:SQUARE:DCYCLE? (see <a href="#">page 1199</a> )	<percent> ::= duty cycle percentage from 20% to 80% in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:MODulation:AM:DEPTH <percent> (see <a href="#">page 1200</a> )	:WGEN<w>:MODulation:AM:DEPTH? (see <a href="#">page 1200</a> )	<percent> ::= AM depth percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:AM:FREQUENCY <frequency> (see <a href="#">page 1201</a> )	:WGEN<w>:MODulation:AM:FREQUENCY? (see <a href="#">page 1201</a> )	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FM:DEVIation <frequency> (see <a href="#">page 1202</a> )	:WGEN<w>:MODulation:FM:DEVIation? (see <a href="#">page 1202</a> )	<frequency> ::= frequency deviation in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FM:FREQUENCY <frequency> (see <a href="#">page 1203</a> )	:WGEN<w>:MODulation:FM:FREQUENCY? (see <a href="#">page 1203</a> )	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FSKEY:FREQUENCY <percent> (see <a href="#">page 1204</a> )	:WGEN<w>:MODulation:FSKEY:FREQUENCY? (see <a href="#">page 1204</a> )	<frequency> ::= hop frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FSKEY:RATE <rate> (see <a href="#">page 1205</a> )	:WGEN<w>:MODulation:FSKEY:RATE? (see <a href="#">page 1205</a> )	<rate> ::= FSK modulation rate in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FUNCtion <shape> (see <a href="#">page 1206</a> )	:WGEN<w>:MODulation:FUNCtion? (see <a href="#">page 1206</a> )	<shape> ::= {SINusoid   SQuare   RAMP} <w> ::= 1 in NR1 format

**Table 70** :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:MODulation:FUNCTion:RAMP:SYMMetry <percent> (see page 1207)	:WGEN<w>:MODulation:FUNCTion:RAMP:SYMMetry? (see <a href="#">page 1207</a> )	<percent> ::= symmetry percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:NOISE <percent> (see page 1208)	:WGEN<w>:MODulation:NOISE? (see <a href="#">page 1208</a> )	<percent> ::= 0 to 100 <w> ::= 1 or 2 in NR1 format
:WGEN<w>:MODulation:STATE {{0   OFF}   {1   ON}} (see page 1209)	:WGEN<w>:MODulation:STATE? (see <a href="#">page 1209</a> )	{0   1} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:TYPE <type> (see page 1210)	:WGEN<w>:MODulation:TYPE? (see <a href="#">page 1210</a> )	<type> ::= {AM   FM   FSK} <w> ::= 1 in NR1 format
:WGEN<w>:OUTPut {{0   OFF}   {1   ON}} (see page 1212)	:WGEN<w>:OUTPut? (see <a href="#">page 1212</a> )	{0   1} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:LOAD <impedance> (see page 1213)	:WGEN<w>:OUTPut:LOAD? (see <a href="#">page 1213</a> )	<impedance> ::= {ONEMeg   FIFTY} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:MODE <mode> (see page 1214)	:WGEN<w>:OUTPut:MODE? (see <a href="#">page 1214</a> )	<mode> ::= {NORMAL   SINGLE}
:WGEN<w>:OUTPut:POLArity <polarity> (see page 1215)	:WGEN<w>:OUTPut:POLArity? (see <a href="#">page 1215</a> )	<polarity> ::= {NORMAL   INVerted} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:SINGLe (see page 1216)	n/a	n/a
:WGEN<w>:PERiod <period> (see page 1217)	:WGEN<w>:PERiod? (see <a href="#">page 1217</a> )	<period> ::= period in seconds in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:RST (see page 1218)	n/a	<w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage <amplitude> (see page 1219)	:WGEN<w>:VOLTage? (see <a href="#">page 1219</a> )	<amplitude> ::= amplitude in volts in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage:HIGH <high> (see page 1220)	:WGEN<w>:VOLTage:HIGH? (see <a href="#">page 1220</a> )	<high> ::= high-level voltage in volts, in NR3 format <w> ::= 1 or 2 in NR1 format

**Table 70** :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:VOLTage:LOW <low> (see <a href="#">page 1221</a> )	:WGEN<w>:VOLTage:LOW? (see <a href="#">page 1221</a> )	<low> ::= low-level voltage in volts, in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage:OFFSet <offset> (see <a href="#">page 1222</a> )	:WGEN<w>:VOLTage:OFFS et? (see <a href="#">page 1222</a> )	<offset> ::= offset in volts in NR3 format <w> ::= 1 or 2 in NR1 format

**Table 71** :WMEMory<r> Commands Summary

Command	Query	Options and Query Returns
:WMEMory<r>:CLEar (see <a href="#">page 1225</a> )	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format
:WMEMory<r>:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 1226</a> )	:WMEMory<r>:DISPlay? (see <a href="#">page 1226</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format {0   1}
:WMEMory<r>:LABel <string> (see <a href="#">page 1227</a> )	:WMEMory<r>:LABel? (see <a href="#">page 1227</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <string> ::= any series of 10 or less ASCII characters enclosed in quotation marks
:WMEMory<r>:SAVE <source> (see <a href="#">page 1228</a> )	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format <source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format NOTE: Only ADD or SUBtract math operations can be saved as reference waveforms.
:WMEMory<r>:SKEW <skew> (see <a href="#">page 1229</a> )	:WMEMory<r>:SKEW? (see <a href="#">page 1229</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <skew> ::= time in seconds in NR3 format
:WMEMory<r>:YOFFset <offset>[suffix] (see <a href="#">page 1230</a> )	:WMEMory<r>:YOFFset? (see <a href="#">page 1230</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <offset> ::= vertical offset value in NR3 format [suffix] ::= {V   mV}

**Table 71** :WMEMory<r> Commands Summary (continued)

Command	Query	Options and Query Returns
:WMEMory<r>:YRANGE <range>[suffix] (see <a href="#">page 1231</a> )	:WMEMory<r>:YRANGE? (see <a href="#">page 1231</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <range> ::= vertical full-scale range value in NR3 format [suffix] ::= {V   mV}
:WMEMory<r>:YScale <scale>[suffix] (see <a href="#">page 1232</a> )	:WMEMory<r>:YScale? (see <a href="#">page 1232</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <scale> ::= vertical units per division value in NR3 format [suffix] ::= {V   mV}

## Syntax Elements

- "[Number Format](#)" on page 174
- "[<NL> \(Line Terminator\)](#)" on page 174
- "[\[ \] \(Optional Syntax Terms\)](#)" on page 174
- "[{ } \(Braces\)](#)" on page 174
- "[::= \(Defined As\)](#)" on page 174
- "[<> \(Angle Brackets\)](#)" on page 175
- "[... \(Ellipsis\)](#)" on page 175
- "[n,...,p \(Value Ranges\)](#)" on page 175
- "[d \(Digits\)](#)" on page 175
- "[Quoted ASCII String](#)" on page 175
- "[Definite-Length Block Response Data](#)" on page 175

### Number Format

NR1 specifies integer data.

NR3 specifies exponential data in floating point format (for example, -1.0E-3).

### <NL> (Line Terminator)

<NL> = new line or linefeed (ASCII decimal 10).

The line terminator, or a leading colon, will send the parser to the "root" of the command tree.

### [ ] (Optional Syntax Terms)

Items enclosed in square brackets, [ ], are optional.

### { } (Braces)

When several items are enclosed by braces, { }, only one of these elements may be selected. Vertical line ( | ) indicates "or". For example, {ON | OFF} indicates that only ON or OFF may be selected, not both.

### ::= (Defined As)

::= means "defined as".

For example, <A> ::= <B> indicates that <A> can be replaced by <B> in any statement containing <A>.

## < > (Angle Brackets)

< > Angle brackets enclose words or characters that symbolize a program code parameter or an interface command.

## ... (Ellipsis)

... An ellipsis (trailing dots) indicates that the preceding element may be repeated one or more times.

## n,...,p (Value Ranges)

n,...,p ::= all integers between n and p inclusive.

## d (Digits)

d ::= A single ASCII numeric character 0 - 9.

## Quoted ASCII String

A quoted ASCII string is a string delimited by either double quotes ("") or single quotes (''). Some command parameters require a quoted ASCII string. For example, when using the Keysight VISA COM library in Visual Basic, the command:

```
myScope.WriteString " :CHANNEL1:LABEL 'One'"
```

has a quoted ASCII string of:

```
'one'
```

In order to read quoted ASCII strings from query return values, some programming languages require special handling or syntax.

## Definite-Length Block Response Data

Definite-length block response data allows any type of device-dependent data to be transmitted over the system interface as a series of 8-bit binary data bytes. This is particularly useful for sending large quantities of data or 8-bit extended ASCII codes. This syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the decimal integer. After the non-zero digit is the decimal integer that states the number of 8-bit data bytes being sent. This is followed by the actual data.

For example, for transmitting 1000 bytes of data, the syntax would be

```
#800001000<1000 bytes of data> <NL>
```

**8** is the number of digits that follow

**00001000** is the number of bytes to be transmitted

**<1000 bytes of data>** is the actual data

# 5 Common (\*) Commands

Commands defined by IEEE 488.2 standard that are common to all instruments.  
See "[Introduction to Common \(\\*\) Commands](#)" on page 180.

**Table 72** Common (\*) Commands Summary

Command	Query	Options and Query Returns																																				
*CLS (see <a href="#">page 181</a> )	n/a	n/a																																				
*ESE <mask> (see <a href="#">page 182</a> )	*ESE? (see <a href="#">page 183</a> )	<p>&lt;mask&gt; ::= 0 to 255; an integer in NR1 format:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>Enables</th> </tr> </thead> <tbody> <tr><td>7</td><td>128</td><td>PON</td><td>Power On</td></tr> <tr><td>6</td><td>64</td><td>URQ</td><td>User Request</td></tr> <tr><td>5</td><td>32</td><td>CME</td><td>Command Error</td></tr> <tr><td>4</td><td>16</td><td>EXE</td><td>Execution Error</td></tr> <tr><td>3</td><td>8</td><td>DDE</td><td>Dev. Dependent Error</td></tr> <tr><td>2</td><td>4</td><td>QYE</td><td>Query Error</td></tr> <tr><td>1</td><td>2</td><td>RQL</td><td>Request Control</td></tr> <tr><td>0</td><td>1</td><td>OPC</td><td>Operation Complete</td></tr> </tbody> </table>	Bit	Weight	Name	Enables	7	128	PON	Power On	6	64	URQ	User Request	5	32	CME	Command Error	4	16	EXE	Execution Error	3	8	DDE	Dev. Dependent Error	2	4	QYE	Query Error	1	2	RQL	Request Control	0	1	OPC	Operation Complete
Bit	Weight	Name	Enables																																			
7	128	PON	Power On																																			
6	64	URQ	User Request																																			
5	32	CME	Command Error																																			
4	16	EXE	Execution Error																																			
3	8	DDE	Dev. Dependent Error																																			
2	4	QYE	Query Error																																			
1	2	RQL	Request Control																																			
0	1	OPC	Operation Complete																																			
n/a	*ESR? (see <a href="#">page 184</a> )	<status> ::= 0 to 255; an integer in NR1 format																																				
n/a	*IDN? (see <a href="#">page 184</a> )	KEYSIGHT TECHNOLOGIES, <model>, <serial number>, X.XX.XX <model> ::= the model number of the instrument <serial number> ::= the serial number of the instrument <X.XX.XX> ::= the software revision of the instrument																																				
n/a	*LRN? (see <a href="#">page 187</a> )	<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format																																				
*OPC (see <a href="#">page 188</a> )	*OPC? (see <a href="#">page 188</a> )	ASCII "1" is placed in the output queue when all pending device operations have completed.																																				

**Table 72** Common (\*) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	*OPT? (see <a href="#">page 189</a> )	<pre> &lt;return_value&gt; ::= 0,0,&lt;license info&gt; &lt;license info&gt; ::= &lt;All field&gt;, &lt;reserved&gt;, &lt;MSO&gt;, &lt;Xilinx FPGA Probe&gt;, &lt;Memory&gt;, &lt;Low Speed Serial&gt;, &lt;Automotive Serial&gt;, &lt;reserved&gt;, &lt;FlexRay Serial&gt;, &lt;Power Measurements&gt;, &lt;RS-232/UART Serial&gt;, &lt;reserved&gt;, &lt;Mask Test&gt;, &lt;reserved&gt;, &lt;Bandwidth&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;I2S Serial&gt;, &lt;reserved&gt;, &lt;Educator's Kit&gt;, &lt;Waveform Generator&gt;, &lt;MIL-1553/ARINC 429 Serial&gt;, &lt;Extended Video&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;Digital Voltmeter&gt;, &lt;Spectrum Visualizer&gt;, &lt;reserved&gt;, &lt;reserved&gt;, &lt;USB 2.0 Low/Full Speed&gt;, &lt;USB 2.0 High Speed&gt; &lt;All field&gt; ::= {0   All} &lt;reserved&gt; ::= 0 &lt;MSO&gt; ::= {0   MSO} &lt;Xilinx FPGA Probe&gt; ::= {0   FPGAX} &lt;Memory&gt; ::= {0   MEMUP} &lt;Low Speed Serial&gt; ::= {0   EMBD} &lt;Automotive Serial&gt; ::= {0   AUTO} &lt;FlexRay Serial&gt; ::= {0   FLEX} &lt;Power Measurements&gt; ::= {0   PWR} &lt;RS-232/UART Serial&gt; ::= {0   COMP} &lt;Mask Test&gt; ::= {0   MASK} &lt;Bandwidth&gt; ::= {0   BW20   BW50} &lt;I2S Serial&gt; ::= {0   AUDIO} &lt;Educator's Kit&gt; ::= {0   EDK} &lt;Waveform Generator&gt; ::= {0   WAVEGEN} &lt;MIL-1553/ARINC 429 Serial&gt; ::= {0   AERO} &lt;Extended Video&gt; ::= {0   VID} &lt;Digital Voltmeter&gt; ::= {0   DVM} &lt;Spectrum Visualizer&gt; ::= {0   SV} &lt;USB 2.0 Low/Full Speed&gt; ::= {0   USF} &lt;USB 2.0 High Speed&gt; ::= {0   HSD} </pre>

**Table 72** Common (\*) Commands Summary (continued)

Command	Query	Options and Query Returns																																				
*RCL <value> (see page 191)	n/a	<value> ::= { 0   1   4   5   6   7   8   9 }																																				
*RST (see page 192)	n/a	See *RST (Reset) (see page 192)																																				
*SAV <value> (see page 195)	n/a	<value> ::= { 0   1   4   5   6   7   8   9 }																																				
*SRE <mask> (see page 196)	*SRE? (see page 197)	<p>&lt;mask&gt; ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. &lt;mask&gt; ::= following values:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>Enables</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER</td> <td>Operation Status Reg</td> </tr> <tr> <td>6</td> <td>64</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB</td> <td>Event Status Bit</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV</td> <td>Message Available</td> </tr> <tr> <td>3</td> <td>8</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG</td> <td>Message</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR</td> <td>User</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG</td> <td>Trigger</td> </tr> </tbody> </table>	Bit	Weight	Name	Enables	7	128	OPER	Operation Status Reg	6	64	---	(Not used.)	5	32	ESB	Event Status Bit	4	16	MAV	Message Available	3	8	---	(Not used.)	2	4	MSG	Message	1	2	USR	User	0	1	TRG	Trigger
Bit	Weight	Name	Enables																																			
7	128	OPER	Operation Status Reg																																			
6	64	---	(Not used.)																																			
5	32	ESB	Event Status Bit																																			
4	16	MAV	Message Available																																			
3	8	---	(Not used.)																																			
2	4	MSG	Message																																			
1	2	USR	User																																			
0	1	TRG	Trigger																																			
n/a	*STB? (see page 198)	<p>&lt;value&gt; ::= 0 to 255; an integer in NR1 format, as shown in the following:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>"1" Indicates</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER</td> <td>Operation status condition occurred.</td> </tr> <tr> <td>6</td> <td>64</td> <td>RQS/</td> <td>Instrument is MSS requesting service.</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB</td> <td>Enabled event status condition occurred.</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV</td> <td>Message available.</td> </tr> <tr> <td>3</td> <td>8</td> <td>---</td> <td>(Not used.)</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG</td> <td>Message displayed.</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR</td> <td>User event condition occurred.</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG</td> <td>A trigger occurred.</td> </tr> </tbody> </table>	Bit	Weight	Name	"1" Indicates	7	128	OPER	Operation status condition occurred.	6	64	RQS/	Instrument is MSS requesting service.	5	32	ESB	Enabled event status condition occurred.	4	16	MAV	Message available.	3	8	---	(Not used.)	2	4	MSG	Message displayed.	1	2	USR	User event condition occurred.	0	1	TRG	A trigger occurred.
Bit	Weight	Name	"1" Indicates																																			
7	128	OPER	Operation status condition occurred.																																			
6	64	RQS/	Instrument is MSS requesting service.																																			
5	32	ESB	Enabled event status condition occurred.																																			
4	16	MAV	Message available.																																			
3	8	---	(Not used.)																																			
2	4	MSG	Message displayed.																																			
1	2	USR	User event condition occurred.																																			
0	1	TRG	A trigger occurred.																																			
*TRG (see page 200)	n/a	n/a																																				
n/a	*TST? (see page 201)	<result> ::= 0 or non-zero value; an integer in NR1 format																																				
*WAI (see page 202)	n/a	n/a																																				

**Introduction to Common (\*) Commands** The common commands are defined by the IEEE 488.2 standard. They are implemented by all instruments that comply with the IEEE 488.2 standard. They provide some of the basic instrument functions, such as instrument identification and reset, reading the instrument setup, and determining how status is read and cleared.

Common commands can be received and processed by the instrument whether they are sent over the interface as separate program messages or within other program messages. If an instrument subsystem has been selected and a common command is received by the instrument, the instrument remains in the selected subsystem. For example, if the program message ":ACQuire:TYPE AVERage; \*CLS; COUNT 256" is received by the instrument, the instrument sets the acquire type, then clears the status information and sets the average count.

In contrast, if a root level command or some other subsystem command is within the program message, you must re-enter the original subsystem after the command. For example, the program message ":ACQuire:TYPE AVERage; :AUToscale; :ACQuire:COUNt 256" sets the acquire type, completes the autoscale, then sets the acquire count. In this example, :ACQuire must be sent again after the :AUToscale command in order to re-enter the ACQuire subsystem and set the count.

**NOTE**

Each of the status registers has an enable (mask) register. By setting the bits in the enable register, you can select the status information you want to use.

## \*CLS (Clear Status)

 (see [page 1334](#))

**Command Syntax** \*CLS

The \*CLS common command clears the status data structures, the device-defined error queue, and the Request-for-OPC flag.

### NOTE

If the \*CLS command immediately follows a program message terminator, the output queue and the MAV (message available) bit are cleared.

### See Also

- "[Introduction to Common \(\\*\) Commands](#)" on page 180
- "[\\*STB \(Read Status Byte\)](#)" on page 198
- "[\\*ESE \(Standard Event Status Enable\)](#)" on page 182
- "[\\*ESR \(Standard Event Status Register\)](#)" on page 184
- "[\\*SRE \(Service Request Enable\)](#)" on page 196
- "[:SYSTem:ERRor](#)" on page 1015

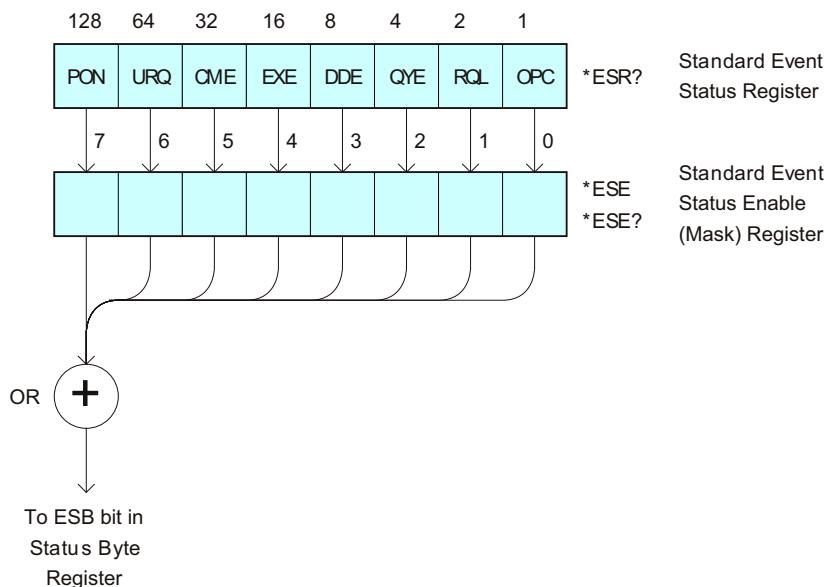
### \*ESE (Standard Event Status Enable)

**C** (see [page 1334](#))

**Command Syntax**    \*ESE <mask\_argument>

<mask\_argument> ::= integer from 0 to 255

The \*ESE common command sets the bits in the Standard Event Status Enable Register. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A "1" in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register. A zero disables the bit.



**Table 73** Standard Event Status Enable (ESE)

Bit	Name	Description	When Set (1 = High = True), Enables:
7	PON	Power On	Event when an OFF to ON transition occurs.
6	URQ	User Request	Event when a front-panel key is pressed.
5	CME	Command Error	Event when a command error is detected.
4	EXE	Execution Error	Event when an execution error is detected.
3	DDE	Device Dependent Error	Event when a device-dependent error is detected.
2	QYE	Query Error	Event when a query error is detected.
1	RQL	Request Control	Event when the device is requesting control. (Not used.)
0	OPC	Operation Complete	Event when an operation is complete.

**Query Syntax** \*ESE?

The \*ESE? query returns the current contents of the Standard Event Status Enable Register.

**Return Format** <mask\_argument><NL>

<mask\_argument> ::= 0, . . . , 255; an integer in NR1 format.

**See Also**

- "[Introduction to Common \(\\*\) Commands](#)" on page 180
- "[\\*ESR \(Standard Event Status Register\)](#)" on page 184
- "[\\*OPC \(Operation Complete\)](#)" on page 188
- "[\\*CLS \(Clear Status\)](#)" on page 181

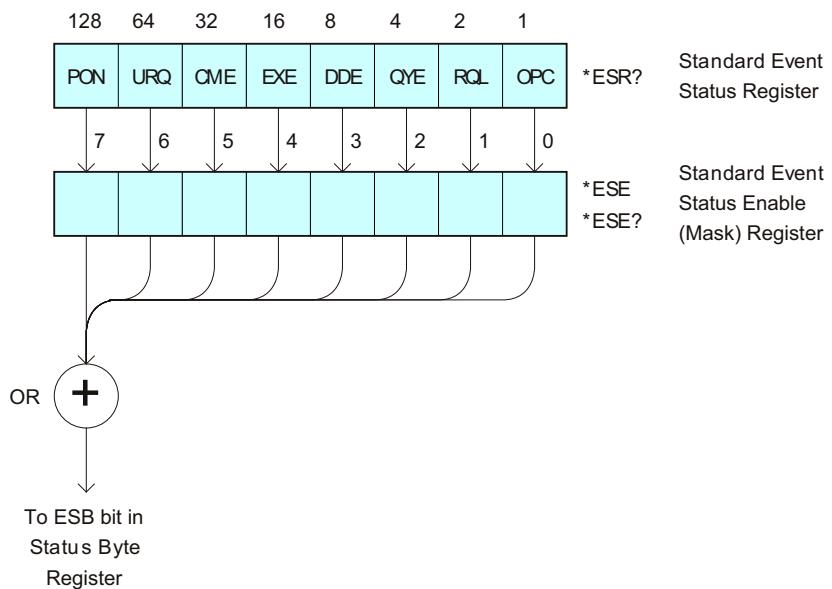
## \*ESR (Standard Event Status Register)

 (see [page 1334](#))

Query Syntax \*ESR?

The \*ESR? query returns the contents of the Standard Event Status Register. When you read the Event Status Register, the value returned is the total bit weights of all of the bits that are high at the time you read the byte. Reading the register clears the Event Status Register.

The following table shows bit weight, name, and condition for each bit.



**Table 74** Standard Event Status Register (ESR)

Bit	Name	Description	When Set (1 = High = True), Indicates:
7	PON	Power On	An OFF to ON transition has occurred.
6	URQ	User Request	A front-panel key has been pressed.
5	CME	Command Error	A command error has been detected.
4	EXE	Execution Error	An execution error has been detected.
3	DDE	Device Dependent Error	A device-dependent error has been detected.
2	QYE	Query Error	A query error has been detected.
1	RQL	Request Control	The device is requesting control. (Not used.)
0	OPC	Operation Complete	Operation is complete.

**Return Format**    <status><NL>  
                  <status> ::= 0, . . . , 255; an integer in NR1 format.

**NOTE**    Reading the Standard Event Status Register clears it. High or 1 indicates the bit is true.

---

- See Also**
- ["Introduction to Common \(\\*\) Commands"](#) on page 180
  - ["\\*ESE \(Standard Event Status Enable\)"](#) on page 182
  - ["\\*OPC \(Operation Complete\)"](#) on page 188
  - ["\\*CLS \(Clear Status\)"](#) on page 181
  - [":SYSTem:ERRor"](#) on page 1015

## \*IDN (Identification Number)

 (see [page 1334](#))

**Query Syntax** \*IDN?

The \*IDN? query identifies the instrument type and software version.

**Return Format**

```
<manufacturer_string>,<model>,<serial_number>,X.XX.XX <NL>
<manufacturer_string> ::= KEYSIGHT TECHNOLOGIES
<model> ::= the model number of the instrument
<serial_number> ::= the serial number of the instrument
X.XX.XX ::= the software revision of the instrument
```

**See Also**

- "[Introduction to Common \(\\*\) Commands](#)" on page 180
- "[\\*OPT \(Option Identification\)](#)" on page 189
- "[":SYSTem:PERSONa\[:MANufacturer\]](#)" on page 1017
- "[":SYSTem:PERSONa\[:MANufacturer\]:DEFault](#)" on page 1018

## \*LRN (Learn Device Setup)



(see [page 1334](#))

### Query Syntax

`*LRN?`

The `*LRN?` query result contains the current state of the instrument. This query is similar to the `:SYST:SETUp?` (see [page 1030](#)) query, except that it contains `":SYST:SET "` before the binary block data. The query result is a valid command that can be used to restore instrument settings at a later time.

### Return Format

```
<learn_string><NL>
<learn_string> ::= :SYST:SET <setup_data>
<setup_data> ::= binary block data in IEEE 488.2 # format
```

`<learn_string>` specifies the current instrument setup. The block size is subject to change with different firmware revisions.

### NOTE

The `*LRN?` query return format has changed from previous Keysight oscilloscopes to match the IEEE 488.2 specification which says that the query result must contain `":SYST:SET "` before the binary block data.

### See Also

- ["Introduction to Common \(\\*\) Commands"](#) on page 180
- ["\\*RCL \(Recall\)"](#) on page 191
- ["\\*SAV \(Save\)"](#) on page 195
- [":SYST:SETUp"](#) on page 1030

## \*OPC (Operation Complete)

 (see [page 1334](#))

**Command Syntax** \*OPC

The \*OPC command sets the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

**Query Syntax** \*OPC?

The \*OPC? query places an ASCII "1" in the output queue when all pending device operations have completed. The interface hangs until this query returns.

**Return Format** <complete><NL>

<complete> ::= 1

**See Also**

- ["Introduction to Common \(\\*\) Commands" on page 180](#)
- ["\\*ESE \(Standard Event Status Enable\)" on page 182](#)
- ["\\*ESR \(Standard Event Status Register\)" on page 184](#)
- ["\\*CLS \(Clear Status\)" on page 181](#)

## \*OPT (Option Identification)

**C** (see [page 1334](#))

**Query Syntax** \*OPT?

The \*OPT? query reports the options installed in the instrument. This query returns a string that identifies the module and its software revision level.

**Return Format** 0,0,<license info>

```

<license info> ::= <All field>, <reserved>, <MSO>, <reserved>,
    <Memory>, <Low Speed Serial>, <Automotive Serial>, <FlexRay Serial>,
    <Power Measurements>, <RS-232/UART Serial>, <Segmented Memory>,
    <Mask Test>, <reserved>, <Bandwidth>, <reserved>, <reserved>,
    <reserved>, <I2S Serial>, <reserved>, <Educator's Kit>,
    <Waveform Generator>, <MIL-1553/ARINC 429 Serial>, <Extended Video>,
    <Advanced Math>, <reserved>, <reserved>, <reserved>, <reserved>,
    <Digital Voltmeter/Counter>, <reserved>, <reserved>, <reserved>,
    <reserved>, <reserved>, <Remote Command Logging>, <reserved>,
    <SENT Serial>, <CAN FD Serial>, <NFC Trigger>

<All field> ::= {0 | All}

<reserved> ::= 0

<MSO> ::= {0 | MSO}

<Memory> ::= {0 | memMax}

<Low Speed Serial> ::= {0 | EMBD}

<Automotive Serial> ::= {0 | AUTO}

<FlexRay Serial> ::= {0 | FLEX}

<Power Measurements> ::= {0 | PWR}

<RS-232/UART Serial> ::= {0 | COMP}

<Segmented Memory> ::= {0 | SGM}

<Mask Test> ::= {0 | MASK}

<Bandwidth> ::= {0 | BW50}

<I2S Serial> ::= {0 | AUDIO}

<Educator's Kit> ::= {0 | EDK}

<Waveform Generator> ::= {0 | WAVEGEN}

<MIL-1553/ARINC 429 Serial> ::= {0 | AERO}

<Extended Video> ::= {0 | VID}

<Advanced Math> ::= {0 | ADVMATH}

<Digital Voltmeter/Counter> ::= {0 | DVMCTR}

<Remote Command Logging> ::= {0 | RML}

```

```
<SENT Serial> ::= { 0 | SENSOR }

<CAN FD Serial> ::= { 0 | CANFD }

<NFC Trigger> ::= { 0 | NFC }
```

The **<MSO>** field indicates whether the unit is a mixed-signal oscilloscope.

The \*OPT? query returns the following:

#### See Also

- ["Introduction to Common \(\\*\) Commands" on page 180](#)
  - ["IDN \(Identification Number\)" on page 186](#)

## \*RCL (Recall)

 (see [page 1334](#))

**Command Syntax**    `*RCL <value>`

`<value> ::= {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}`

The \*RCL command restores the state of the instrument from the specified save/recall register.

**See Also**

- ["Introduction to Common \(\\*\) Commands"](#) on page 180
- ["\\*SAV \(Save\)"](#) on page 195

**\*RST (Reset)**

 (see [page 1334](#))

**Command Syntax**    **\*RST**

The \*RST command places the instrument in a known state. This is the same as pressing **[Save/Recall] > Default/Erase > Factory Default** on the front panel.

When you perform a factory default setup, there are no user settings that remain unchanged. To perform the equivalent of the front panel's **[Default Setup]** key, where some user settings (like preferences) remain unchanged, use the :SYSTem:PRESet command.

Reset conditions are:

Acquire Menu	
Mode	Normal
Averaging	Off
# Averages	8

Analog Channel Menu	
Channel 1	On
Channel 2	Off
Volts/division	5.00 V
Offset	0.00
Coupling	DC
Probe attenuation	AutoProbe (if AutoProbe is connected), otherwise 1.0:1
Vernier	Off
Invert	Off
BW limit	Off
Impedance	1 M Ohm
Units	Volts
Skew	0

Cursor Menu	
Source	Channel 1

<b>Digital Channel Menu (MSO models only)</b>	
Channel 0 - 15	Off
Labels	Off
Threshold	TTL (1.4 V)

<b>Display Menu</b>	
Persistence	Off
Grid	20%

<b>Quick Meas Menu</b>	
Source	Channel 1

<b>Run Control</b>	
	Scope is running

<b>Time Base Menu</b>	
Main time/division	100 us
Main time base delay	0.00 s
Delay time/division	500 ns
Delay time base delay	0.00 s
Reference	center
Mode	main
Vernier	Off

<b>Trigger Menu</b>	
Type	Edge
Mode	Auto
Coupling	dc
Source	Channel 1
Level	0.0 V
Slope	Positive

Trigger Menu	
HF Reject and noise reject	Off
Holdoff	40 ns
External probe attenuation	10:1
External Units	Volts
External Impedance	1 M Ohm (cannot be changed)

- See Also**
- "[Introduction to Common \(\\*\) Commands](#)" on page 180
  - "[":SYSTem:PRESet](#)" on page 1019

**Example Code**

```

' RESET - This command puts the oscilloscope into a known state.
' This statement is very important for programs to work as expected.
' Most of the following initialization commands are initialized by
' *RST. It is not necessary to reinitialize them unless the default
' setting is not suitable for your application.
myScope.WriteString "*RST" ' Reset the oscilloscope to the defaults.

```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

**\*SAV (Save)** (see [page 1334](#))**Command Syntax**    **\*SAV <value>****<value>** ::= { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }

The \*SAV command stores the current state of the instrument in a save register. The data parameter specifies the register where the data will be saved.

**See Also**

- ["Introduction to Common \(\\*\) Commands"](#) on page 180
- ["\\*RCL \(Recall\)"](#) on page 191

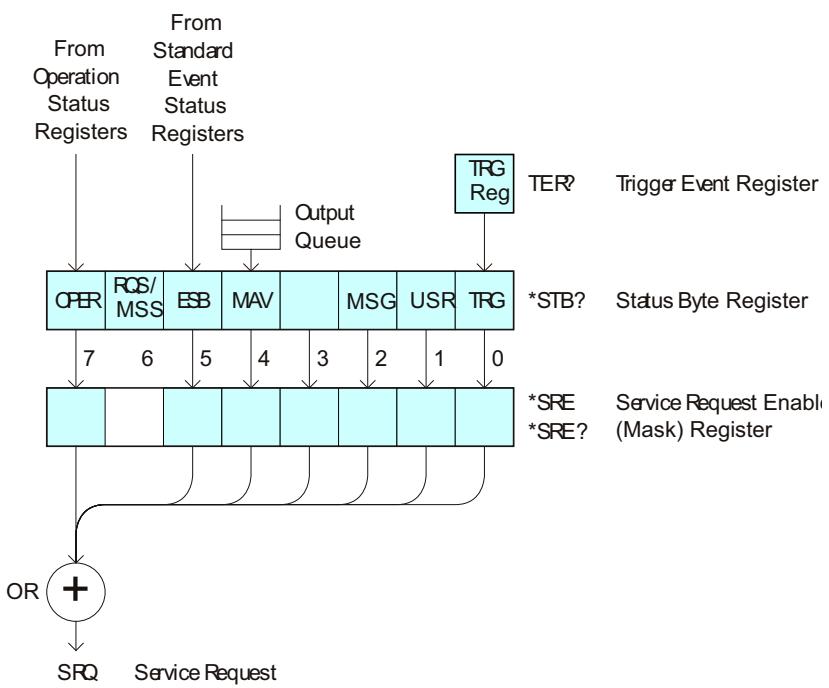
## \*SRE (Service Request Enable)

 (see [page 1334](#))

**Command Syntax**    \*SRE <mask>

<mask> ::= integer with values defined in the following table.

The \*SRE command sets the bits in the Service Request Enable Register. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A one in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A zero disables the bit.



**Table 75** Service Request Enable Register (SRE)

Bit	Name	Description	When Set (1 = High = True), Enables:
7	OPER	Operation Status Register	Interrupts when enabled conditions in the Operation Status Register (OPER) occur.
6	---	---	(Not used.)
5	ESB	Event Status Bit	Interrupts when enabled conditions in the Standard Event Status Register (ESR) occur.
4	MAV	Message Available	Interrupts when messages are in the Output Queue.
3	---	---	(Not used.)
2	MSG	Message	Interrupts when an advisory has been displayed on the oscilloscope.
1	USR	User Event	Interrupts when enabled user event conditions occur.
0	TRG	Trigger	Interrupts when a trigger occurs.

**Query Syntax** \*SRE?

The \*SRE? query returns the current value of the Service Request Enable Register.

**Return Format** <mask><NL>

<mask> ::= sum of all bits that are set, 0,...,255;  
an integer in NR1 format

**See Also**

- ["Introduction to Common \(\\*\) Commands"](#) on page 180
- ["\\*STB \(Read Status Byte\)"](#) on page 198
- ["\\*CLS \(Clear Status\)"](#) on page 181

**\*STB (Read Status Byte)**

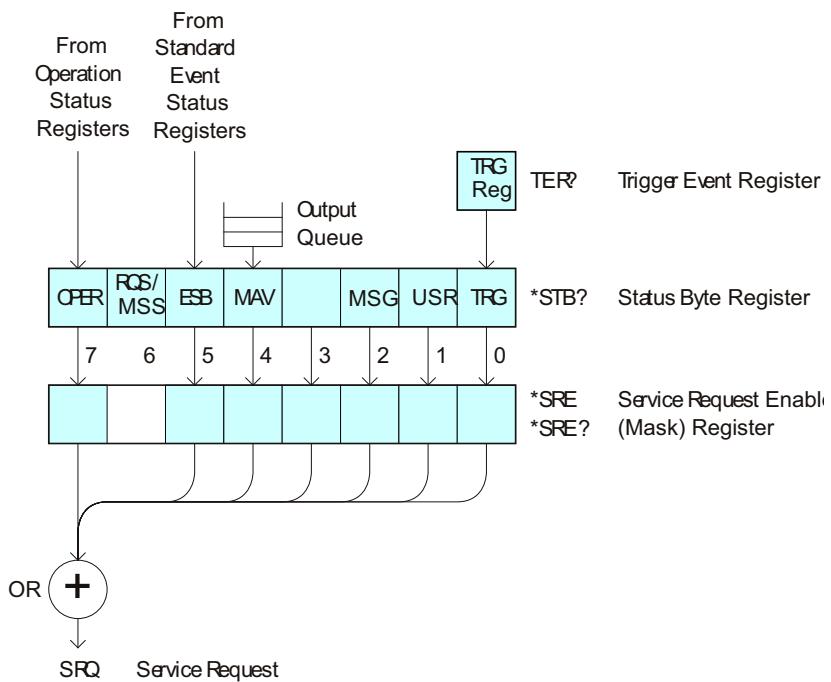
**C** (see [page 1334](#))

**Query Syntax** \*STB?

The \*STB? query returns the current value of the instrument's status byte. The MSS (Master Summary Status) bit is reported on bit 6 instead of the RQS (request service) bit. The MSS indicates whether or not the device has at least one reason for requesting service.

**Return Format** <value><NL>

<value> ::= 0, ..., 255; an integer in NR1 format



**Table 76** Status Byte Register (STB)

Bit	Name	Description	When Set (1 = High = True), Indicates:
7	OPER	Operation Status Register	An enabled condition in the Operation Status Register (OPER) has occurred.
6	RQS	Request Service	When polled, that the device is requesting service.
	MSS	Master Summary Status	When read (by *STB?), whether the device has a reason for requesting service.
5	ESB	Event Status Bit	An enabled condition in the Standard Event Status Register (ESR) has occurred.
4	MAV	Message Available	There are messages in the Output Queue.
3	---	---	(Not used, always 0.)
2	MSG	Message	An advisory has been displayed on the oscilloscope.
1	USR	User Event	An enabled user event condition has occurred.
0	TRG	Trigger	A trigger has occurred.

**NOTE**

To read the instrument's status byte with RQS reported on bit 6, use the interface Serial Poll.

**See Also**

- "[Introduction to Common \(\\*\) Commands](#)" on page 180
- "["\\*SRE \(Service Request Enable\)](#)" on page 196

## \*TRG (Trigger)

 (see [page 1334](#))

**Command Syntax** \*TRG

The \*TRG command has the same effect as the :DIGITIZE command with no parameters.

**See Also**

- "[Introduction to Common \(\\*\) Commands](#)" on page 180
- "[:DIGITIZE](#)" on page 215
- "[:RUN](#)" on page 236
- "[:STOP](#)" on page 240

**\*TST (Self Test)**(see [page 1334](#))**Query Syntax** \*TST?

The \*TST? query performs a self-test on the instrument. The result of the test is placed in the output queue. A zero indicates the test passed and a non-zero indicates the test failed. If the test fails, refer to the troubleshooting section of the *Service Guide*.

**Return Format** <result><NL>

<result> ::= 0 or non-zero value; an integer in NR1 format

**See Also** · "Introduction to Common (\*) Commands" on page 180

## \*WAI (Wait To Continue)

 (see [page 1334](#))

**Command Syntax** \*WAI

The \*WAI command has no function in the oscilloscope, but is parsed for compatibility with other instruments.

**See Also** • ["Introduction to Common \(\\*\) Commands"](#) on page 180

# 6 Root (:) Commands

Control many of the basic functions of the oscilloscope and reside at the root level of the command tree. See "[Introduction to Root \(:\) Commands](#)" on page 206.

**Table 77** Root (:) Commands Summary

Command	Query	Options and Query Returns
:ACTivity (see <a href="#">page 207</a> )	:ACTivity? (see <a href="#">page 207</a> )	<return value> ::= <edges>,<levels> <edges> ::= presence of edges (32-bit integer in NR1 format) <levels> ::= logical highs or lows (32-bit integer in NR1 format)
n/a	:AER? (see <a href="#">page 208</a> )	{0   1}; an integer in NR1 format
:AUToscale [ <a href="#">source</a> ] [, . . . , <a href="#">source</a> ] (see <a href="#">page 209</a> )	n/a	<source> ::= CHANnel<n> for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   POD1   POD2} for MSO models <source> can be repeated up to 5 times <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:AUToscale:AMODE <value> (see <a href="#">page 211</a> )	:AUToscale:AMODE? (see <a href="#">page 211</a> )	<value> ::= {NORMal   CURRent}
:AUToscale:CHANnels <value> (see <a href="#">page 212</a> )	:AUToscale:CHANnels? (see <a href="#">page 212</a> )	<value> ::= {ALL   DISPlayed}
:AUToscale:FDEBug {{0   OFF}   {1   ON}} (see <a href="#">page 213</a> )	:AUToscale:FDEBug? (see <a href="#">page 213</a> )	{0   1}

**Table 77** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:BLANK [<source>] (see <a href="#">page 214</a> )	n/a	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;}   DIGItal&lt;d&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p>
:DIGItize [<source>[, . . . , <source>]] (see <a href="#">page 215</a> )	n/a	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;}   DIGItal&lt;d&gt;   POD{1   2}   BUS{1   2}   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS{1   2}} for MSO models</p> <p>&lt;source&gt; can be repeated up to 5 times</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:HWEenable <n> (see <a href="#">page 217</a> )	:HWEenable? (see <a href="#">page 217</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:HWERegister:CONDiti on? (see <a href="#">page 219</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:HWERegister[:EVENT]? (see <a href="#">page 220</a> )	<n> ::= 16-bit integer in NR1 format
:MTEenable <n> (see <a href="#">page 221</a> )	:MTEenable? (see <a href="#">page 221</a> )	<n> ::= 16-bit integer in NR1 format
n/a	:MTERegister[:EVENT]? (see <a href="#">page 223</a> )	<n> ::= 16-bit integer in NR1 format

**Table 77** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns																																				
:OPEE <n> (see page 225)	:OPEE? (see page 226)	<n> ::= 15-bit integer in NR1 format																																				
n/a	:OPERegister:CONDiti on? (see page 227)	<n> ::= 15-bit integer in NR1 format																																				
n/a	:OPERegister[:EVENT]? (see page 229)	<n> ::= 15-bit integer in NR1 format																																				
:OVLenable <mask> (see page 231)	:OVLenable? (see page 232)	<p>&lt;mask&gt; ::= 16-bit integer in NR1 format as shown:</p> <table> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Input</th> </tr> </thead> <tbody> <tr> <td>---</td> <td>-----</td> <td>-----</td> </tr> <tr> <td>10</td> <td>1024</td> <td>Ext Trigger Fault</td> </tr> <tr> <td>9</td> <td>512</td> <td>Channel 4 Fault</td> </tr> <tr> <td>8</td> <td>256</td> <td>Channel 3 Fault</td> </tr> <tr> <td>7</td> <td>128</td> <td>Channel 2 Fault</td> </tr> <tr> <td>6</td> <td>64</td> <td>Channel 1 Fault</td> </tr> <tr> <td>4</td> <td>16</td> <td>Ext Trigger OVL</td> </tr> <tr> <td>3</td> <td>8</td> <td>Channel 4 OVL</td> </tr> <tr> <td>2</td> <td>4</td> <td>Channel 3 OVL</td> </tr> <tr> <td>1</td> <td>2</td> <td>Channel 2 OVL</td> </tr> <tr> <td>0</td> <td>1</td> <td>Channel 1 OVL</td> </tr> </tbody> </table>	Bit	Weight	Input	---	-----	-----	10	1024	Ext Trigger Fault	9	512	Channel 4 Fault	8	256	Channel 3 Fault	7	128	Channel 2 Fault	6	64	Channel 1 Fault	4	16	Ext Trigger OVL	3	8	Channel 4 OVL	2	4	Channel 3 OVL	1	2	Channel 2 OVL	0	1	Channel 1 OVL
Bit	Weight	Input																																				
---	-----	-----																																				
10	1024	Ext Trigger Fault																																				
9	512	Channel 4 Fault																																				
8	256	Channel 3 Fault																																				
7	128	Channel 2 Fault																																				
6	64	Channel 1 Fault																																				
4	16	Ext Trigger OVL																																				
3	8	Channel 4 OVL																																				
2	4	Channel 3 OVL																																				
1	2	Channel 2 OVL																																				
0	1	Channel 1 OVL																																				
n/a	:OVLRegister? (see page 233)	<value> ::= integer in NR1 format. See OVLenable for <value>																																				
:PRINT [<options>] (see page 235)	n/a	<p>&lt;options&gt; ::= [&lt;print option&gt;] [, . . . , &lt;print option&gt;]</p> <p>&lt;print option&gt; ::= {COLOR   GRAYscale   PRINTER0   BMP8bit   BMP   PNG   NOFactors   FACTors}</p> <p>&lt;print option&gt; can be repeated up to 5 times.</p>																																				
:RUN (see page 236)	n/a	n/a																																				
n/a	:SERial (see page 237)	<return value> ::= unquoted string containing serial number																																				
:SINGle (see page 238)	n/a	n/a																																				

**Table 77** Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:STATus? <display> (see <a href="#">page 239</a> )	{0   1} <display> ::= {CHANnel<n>   DIGItal<d>   POD{1   2}   BUS{1   2}   FUNCtion<m>   MATH<m>   FFT   SBUS{1   2}   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format
:STOP (see <a href="#">page 240</a> )	n/a	n/a
n/a	:TER? (see <a href="#">page 241</a> )	{0   1}
:VIEW <source> (see <a href="#">page 242</a> )	n/a	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   FFT   SBUS{1   2}   WMEMORY<r>} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   POD{1   2}   BUS{1   2}   FUNCtion<m>   MATH<m>   FFT   SBUS{1   2}   WMEMORY<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format

**Introduction to Root (:) Commands** Root level commands control many of the basic operations of the instrument. These commands are always recognized by the parser if they are prefixed with a colon, regardless of current command tree position. After executing a root-level command, the parser is positioned at the root of the command tree.

**:ACTivity**

**N** (see [page 1334](#))

**Command Syntax** :ACTivity

The :ACTivity command clears the cumulative edge variables for the next activity query.

**Query Syntax** :ACTivity?

The :ACTivity? query returns whether there has been activity (edges) on the digital channels since the last query, and returns the current logic levels.

**NOTE**

Because the :ACTivity? query returns edge activity since the last :ACTivity? query, you must send this query twice before the edge activity result is valid.

**Return Format**

```
<edges>,<levels><NL>
<edges> ::= presence of edges (16-bit integer in NR1 format).
<levels> ::= logical highs or lows (16-bit integer in NR1 format).
bit 0 ::= DIGital 0
bit 15 ::= DIGital 15
```

**NOTE**

A bit = 0 (zero) in the <edges> result indicates that no edges were detected on that channel (across the specified threshold voltage) since the last query.

A bit = 1 (one) in the <edges> result indicates that edges have been detected on that channel (across the specified threshold voltage) since the last query.

(The threshold voltage must be set appropriately for the logic levels of the signals being probed.)

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":POD<n>:THreshold](#)" on page 605
- "[":DIGital<d>:THreshold](#)" on page 327

**:AER (Arm Event Register)**(see [page 1334](#))**Query Syntax**`:AER?`

The AER query reads the Arm Event Register. After the Arm Event Register is read, it is cleared. A "1" indicates the trigger system is in the armed state, ready to accept a trigger.

The Armed Event Register is summarized in the Wait Trig bit of the Operation Status Event Register. A Service Request can be generated when the Wait Trig bit transitions and the appropriate enable bits have been set in the Operation Status Enable Register (OPEE) and the Service Request Enable Register (SRE).

**Return Format**

```
<value><NL>
<value> ::= {0 | 1}; an integer in NR1 format.
```

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[:OPEE \(Operation Status Enable Register\)](#)" on page 225
- "[:OPERRegister:CONDition \(Operation Status Condition Register\)](#)" on page 227
- "[:OPERRegister\[:EVENT\] \(Operation Status Event Register\)](#)" on page 229
- "[\\*STB \(Read Status Byte\)](#)" on page 198
- "[\\*SRE \(Service Request Enable\)](#)" on page 196

## :AUToscale

 (see [page 1334](#))

### Command Syntax

```
:AUToscale
:AUToscale [<source>[,...<source>]]
<source> ::= CHANnel<n> for the DSO models
<source> ::= {DIGItal<d> | POD1 | POD2 | CHANnel<n>} for the
MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
The <source> parameter may be repeated up to 5 times.
```

The :AUToscale command evaluates all input signals and sets the correct conditions to display the signals. This is the same as pressing the **[Auto Scale]** key on the front panel.

If one or more sources are specified, those specified sources will be enabled and all others blanked. The autoscale channels mode (see "[:AUToscale:CHANnels](#)" on page 212) is set to DISPlayed channels. Then, the autoscale is performed.

When the :AUToscale command is sent, the following conditions are affected and actions are taken:

- Thresholds.
- Channels with activity around the trigger point are turned on, others are turned off.
- Channels are reordered on screen; analog channel 1 first, followed by the remaining analog channels, then the digital channels 0-15.
- Delay is set to 0 seconds.
- Time/Div.

The :AUToscale command does not affect the following conditions:

- Label names.
- Trigger conditioning.

The :AUToscale command turns off the following items:

- Cursors.
- Measurements.
- Math waveforms.
- Reference waveforms.
- Zoomed (delayed) time base mode.

For further information on :AUToscale, see the *User's Guide*.

<b>See Also</b>	<ul style="list-style-type: none"><li>· "<a href="#">Introduction to Root (:) Commands</a>" on page 206</li><li>· "<a href="#">:AUToscale:CHANnels</a>" on page 212</li><li>· "<a href="#">:AUToscale:AMODE</a>" on page 211</li></ul>
<b>Example Code</b>	<pre>' AUTOSCALE - This command evaluates all the input signals and sets ' the correct conditions to display all of the active signals. myScope.WriteString ":AUToscale"      ' Same as pressing Auto Scale key.</pre> <p>See complete example programs at: <a href="#">Chapter 42, “Programming Examples,”</a> starting on page 1343</p>

**:AUToscale:AMODE****N** (see [page 1334](#))

**Command Syntax**    `:AUToscale:AMODE <value>`  
                  `<value> ::= {NORMAL | CURR}`

The :AUToscale:AMODE command specifies the acquisition mode that is set by subsequent :AUToscales.

- When NORMAL is selected, an :AUToscale command sets the NORMAL acquisition type and the RTIMe (real-time) acquisition mode.
- When CURR is selected, the current acquisition type and mode are kept on subsequent :AUToscales.

Use the :ACQuire:TYPE and :ACQuire:MODE commands to set the acquisition type and mode.

**Query Syntax**    `:AUToscale:AMODE?`

The :AUToscale:AMODE? query returns the autoscale acquire mode setting.

**Return Format**    `<value><NL>`  
                  `<value> ::= {NORM | CURR}`

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":AUToscale"](#)" on page 209
- "[":AUToscale:CHANnels"](#)" on page 212
- "[":ACQuire:TYPE"](#)" on page 255
- "[":ACQuire:MODE"](#)" on page 247

**:AUToscale:CHANnels****N** (see [page 1334](#))

**Command Syntax**    `:AUToscale:CHANnels <value>`  
`<value> ::= {ALL | DISPlayed}`

The :AUToscale:CHANnels command specifies which channels will be displayed on subsequent :AUToscales.

- When ALL is selected, all channels that meet the requirements of :AUToscale will be displayed.
- When DISPlayed is selected, only the channels that are turned on are autoscaled.

Use the :VIEW or :BLANK root commands to turn channels on or off.

**Query Syntax**    `:AUToscale:CHANnels?`

The :AUToscale:CHANnels? query returns the autoscale channels setting.

**Return Format**    `<value><NL>`  
`<value> ::= {ALL | DISP}`

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":AUToscale"](#) on page 209
- "[":AUToscale:AMODE"](#) on page 211
- "[":VIEW"](#) on page 242
- "[":BLANK"](#) on page 214

## :AUToscale:FDEBug

**N** (see [page 1334](#))

**Command Syntax** :AUToscale:FDEBug <on\_off>

<on\_off> ::= {{1 | ON} | {0 | OFF}}

The :AUToscale:FDEBug command turns fast debug auto scaling on or off.

The Fast Debug option changes the behavior of :AUToscale to let you make quick visual comparisons to determine whether the signal being probed is a DC voltage, ground, or an active AC signal.

Channel coupling is maintained for easy viewing of oscillating signals.

**Query Syntax** :AUToscale:FDEBug?

The :AUToscale:FDEBug? query returns the current autoscale fast debug setting.

**Return Format** <on\_off><NL>

<on\_off> ::= {1 | 0}

**See Also** • ["Introduction to Root \(:\) Commands" on page 206](#)  
• [":AUToscale" on page 209](#)

**:BLANK**

**N** (see [page 1334](#))

**Command Syntax**

```
:BLANK [<source>]

<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for the DSO models

<source> ::= {CHANnel<n> | DIGItal<d> | POD{1 | 2}
              | BUS{1 | 2} | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for the MSO models

<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :BLANK command turns off (stops displaying) the specified channel, digital pod, math function, serial decode bus, or reference waveform location. The :BLANK command with no parameter turns off all sources.

**NOTE**

To turn on (start displaying) a channel, etc., use the :VIEW command. The DISPlay commands, :CHANnel<n>:DISPlay, :FUNCtion:DISPlay, :POD<n>:DISPlay, :DIGItal<n>:DISPlay, :SBUS<n>:DISPlay, or :WMEMory<r>:DISPlay, are the preferred method to turn on/off a channel, etc.

**NOTE**

MATH<m> is an alias for FUNCtion<m>.

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":DISPlay:CLEar](#)" on page 338
- "[":CHANnel<n>:DISPlay](#)" on page 284
- "[":DIGItal<d>:DISPlay](#)" on page 323
- "[":FUNCtion<m>:DISPlay](#)" on page 391
- "[":POD<n>:DISPlay](#)" on page 603
- "[":SBUS<n>:DISPlay](#)" on page 726
- "[":WMEMory<r>:DISPlay](#)" on page 1226
- "[":STATUs](#)" on page 239
- "[":VIEW](#)" on page 242

**Example Code**

- "[Example Code](#)" on page 242

## :DIGitize

**C** (see [page 1334](#))

**Command Syntax**

```
:DIGITIZE [<source>[, . . . , <source>]]  
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}}  
for the DSO models  
<source> ::= {CHANnel<n> | DIGItal<d> | POD{1 | 2}  
| BUS{1 | 2} | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}}  
for the MSO models  
<n> ::= 1 to (# analog channels) in NR1 format  
<d> ::= 0 to (# digital channels - 1) in NR1 format  
<m> ::= 1 to (# math functions) in NR1 format  
The <source> parameter may be repeated up to 5 times.
```

The :DIGITIZE command is a specialized RUN command. It causes the instrument to acquire waveforms according to the settings of the :ACQuire commands subsystem. When the acquisition is complete, the instrument is stopped.

If no argument is given, :DIGITIZE acquires the channels currently displayed. If no channels are displayed, all channels are acquired.

### NOTE

The :DIGITIZE command is only executed when the :TIMEbase:MODE is MAIN or WINDOW.

### NOTE

To halt a :DIGITIZE in progress, use the device clear command.

### NOTE

MATH<m> is an alias for FUNCtion<m>.

### See Also

- ["Introduction to Root \(:\) Commands"](#) on page 206
- [":RUN"](#) on page 236
- [":SINGle"](#) on page 238
- [":STOP"](#) on page 240
- [":TIMEbase:MODE"](#) on page 1037
- [Chapter 7](#), “:ACQuire Commands,” starting on page 243
- [Chapter 34](#), “:WAVEform Commands,” starting on page 1145

**Example Code**

```
' Capture an acquisition using :DIGitize.  
' -----  
myScope.WriteString ":DIGitize CHANnel1"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

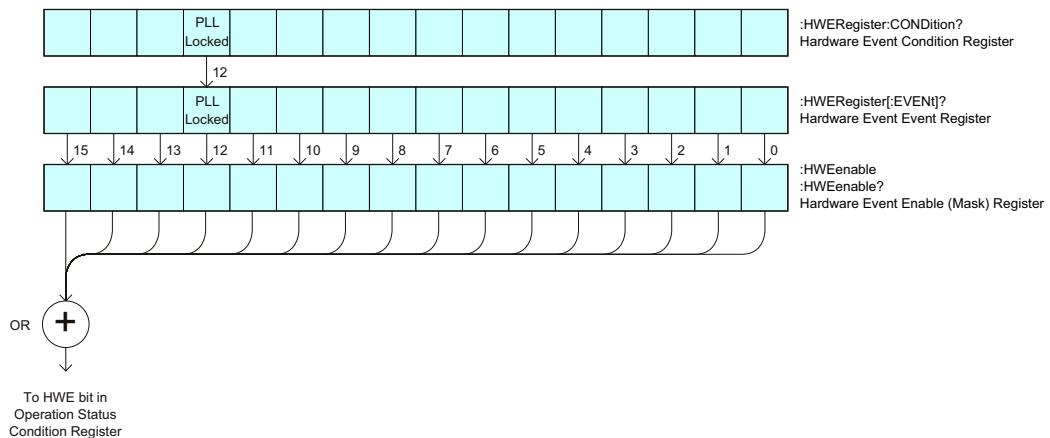
## :HWEenable (Hardware Event Enable Register)

**N** (see [page 1334](#))

**Command Syntax**

```
:HWEenable <mask>
<mask> ::= 16-bit integer
```

The :HWEenable command sets a mask in the Hardware Event Enable register. Set any of the following bits to "1" to enable bit 12 in the Operation Status Condition Register and potentially cause an SRQ (Service Request interrupt) to be generated.



**Table 78** Hardware Event Enable Register (HWEenable)

Bit	Name	Description	When Set (1 = High = True), Enables:
15-13	---	---	(Not used.)
12	PLL Locked	PLL Locked	This bit is for internal use and is not intended for general use.
11-0	---	---	(Not used.)

**Query Syntax**

```
:HWEenable?
```

The :HWEenable? query returns the current value contained in the Hardware Event Enable register as an integer number.

**Return Format**

```
<value><NL>
<value> ::= integer in NR1 format.
```

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[:AER \(Arm Event Register\)](#)" on page 208
- "[:CHANnel<n>:PROtection](#)" on page 295
- "[:OPERegister:\[EVENT\] \(Operation Status Event Register\)](#)" on page 229

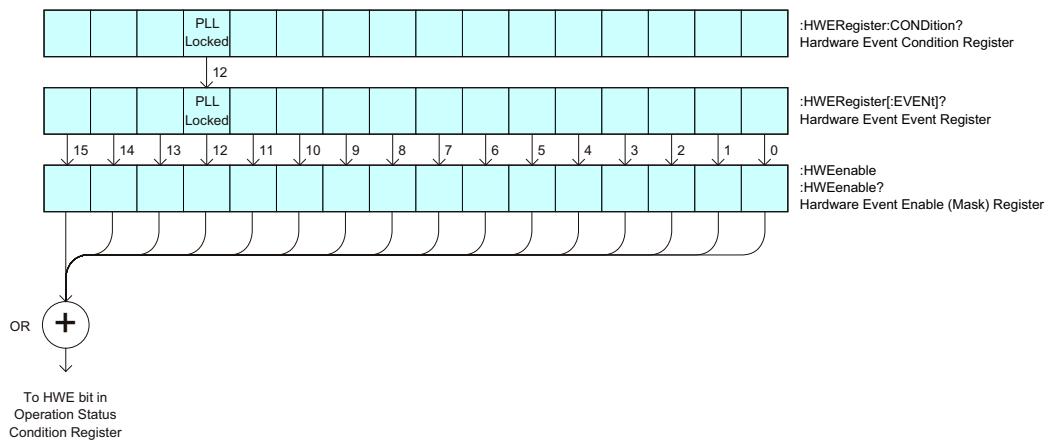
- "[":OVLenable \(Overload Event Enable Register\)](#)" on page 231
- "[":OVLregister \(Overload Event Register\)](#)" on page 233
- "[":\\*STB \(Read Status Byte\)](#)" on page 198
- "[":\\*SRE \(Service Request Enable\)](#)" on page 196

## :HWERegister:CONDition (Hardware Event Condition Register)

**N** (see page 1334)

**Query Syntax** :HWERegister:CONDition?

The :HWERegister:CONDition? query returns the integer value contained in the Hardware Event Condition Register.



**Table 79** Hardware Event Condition Register

Bit	Name	Description	When Set (1 = High = True), Indicates:
15-13	---	---	(Not used.)
12	PLL Locked	PLL Locked	This bit is for internal use and is not intended for general use.
11-0	---	---	(Not used.)

**Return Format** <value><NL>

<value> ::= integer in NR1 format.

**See Also**

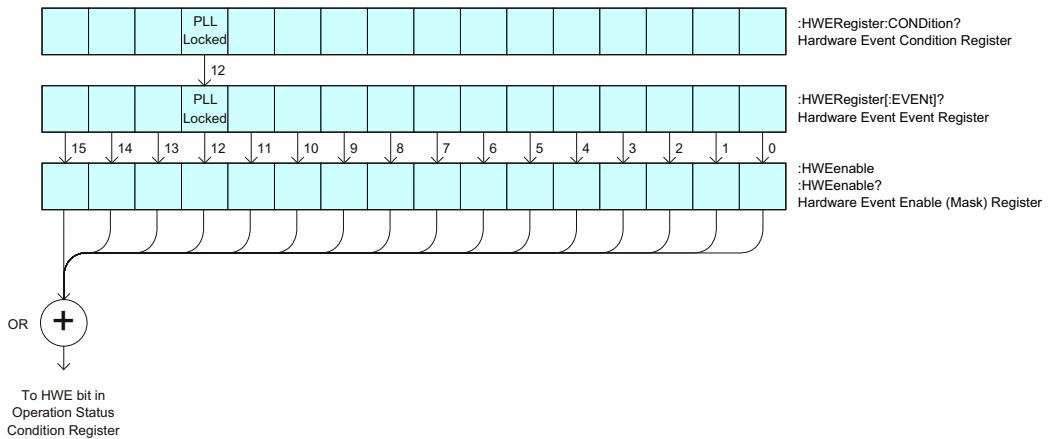
- "Introduction to Root (:) Commands" on page 206
- ".CHANnel<n>:PROTection" on page 295
- ".:OPEE (Operation Status Enable Register)" on page 225
- ".:OPERegister[:EVENT] (Operation Status Event Register)" on page 229
- ".:OVLenable (Overload Event Enable Register)" on page 231
- ".:OVLRegister (Overload Event Register)" on page 233
- ".\*:STB (Read Status Byte)" on page 198
- ".\*:SRE (Service Request Enable)" on page 196

## :HWERegister[:EVENT] (Hardware Event Event Register)

**N** (see [page 1334](#))

**Query Syntax** :HWERegister [:EVENT] ?

The :HWERegister[:EVENT]? query returns the integer value contained in the Hardware Event Event Register.



**Table 80** Hardware Event Event Register

Bit	Name	Description	When Set (1 = High = True), Indicates:
15-13	---	---	(Not used.)
12	PLL Locked	PLL Locked	This bit is for internal use and is not intended for general use.
11-0	---	---	(Not used.)

**Return Format** <value><NL>

<value> ::= integer in NR1 format.

**See Also**

- "[Introduction to Root \(: Commands](#)" on page 206
- "[":CHANnel<n>:PROTection](#)" on page 295
- "[":OPEE \(Operation Status Enable Register\)"](#) on page 225
- "[":OPERegister:CONDition \(Operation Status Condition Register\)"](#) on page 227
- "[":OVLenable \(Overload Event Enable Register\)"](#) on page 231
- "[":OVLRegister \(Overload Event Register\)"](#) on page 233
- "[":\\*STB \(Read Status Byte\)"](#) on page 198
- "[":\\*SRE \(Service Request Enable\)"](#) on page 196

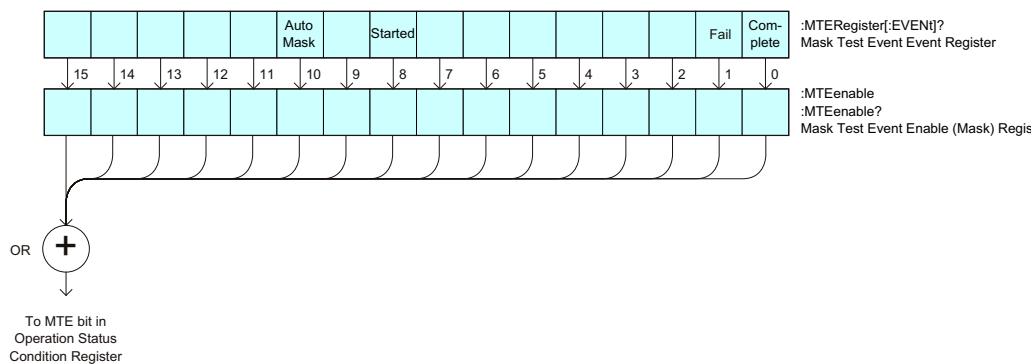
## :MTEenable (Mask Test Event Enable Register)

**N** (see [page 1334](#))

**Command Syntax**

```
:MTEenable <mask>
<mask> ::= 16-bit integer
```

The :MTEenable command sets a mask in the Mask Test Event Enable register. Set any of the following bits to "1" to enable bit 9 in the Operation Status Condition Register and potentially cause an SRQ (Service Request) interrupt to be generated.



**Table 81** Mask Test Event Enable Register (MTEenable)

Bit	Name	Description	When Set (1 = High = True), Enables:
15-11	---	---	(Not used.)
10	Auto Mask	Auto Mask Created	Auto mask creation completed.
9	---	---	(Not used.)
8	Started	Mask Testing Started	Mask testing started.
7-2	---	---	(Not used.)
1	Fail	Mask Test Fail	Mask test failed.
0	Complete	Mask Test Complete	Mask test is complete.

**Query Syntax**

```
:MTEenable?
```

The :MTEenable? query returns the current value contained in the Mask Test Event Enable register as an integer number.

**Return Format**

```
<value><NL>
<value> ::= integer in NR1 format.
```

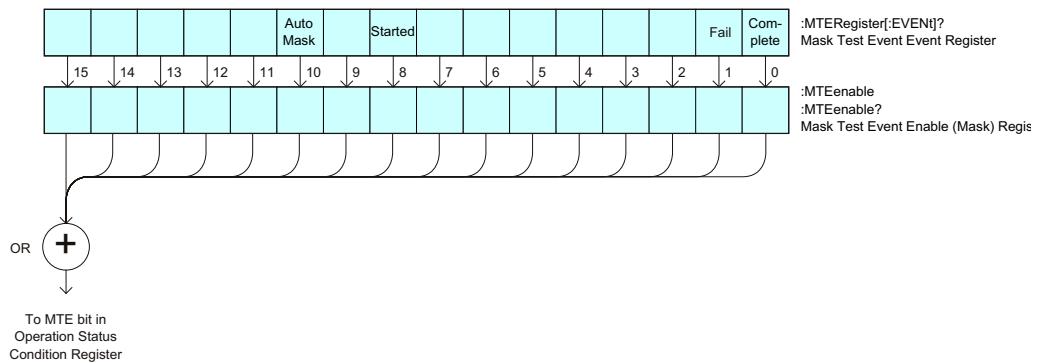
- See Also**
- "[Introduction to Root \(:\) Commands](#)" on page 206
  - "[:AER \(Arm Event Register\)](#)" on page 208
  - "[:CHANnel<n>:PROTection](#)" on page 295
  - "[:OPERegister\[:EVENT\] \(Operation Status Event Register\)](#)" on page 229
  - "[:OVLenable \(Overload Event Enable Register\)](#)" on page 231
  - "[:OVLRegister \(Overload Event Register\)](#)" on page 233
  - "[:\\*STB \(Read Status Byte\)](#)" on page 198
  - "[:\\*SRE \(Service Request Enable\)](#)" on page 196

## :MTERegister[:EVENT] (Mask Test Event Event Register)

**N** (see [page 1334](#))

**Query Syntax** :MTERegister [:EVENT] ?

The :MTERegister[:EVENT]? query returns the integer value contained in the Mask Test Event Event Register and clears the register.



**Table 82** Mask Test Event Event Register

Bit	Name	Description	When Set (1 = High = True), Indicates:
15-11	---	---	(Not used.)
10	Auto Mask	Auto Mask Created	Auto mask creation completed.
9	---	---	(Not used.)
8	Started	Mask Testing Started	Mask testing started.
7-2	---	---	(Not used.)
1	Fail	Mask Test Fail	The mask test failed.
0	Complete	Mask Test Complete	The mask test is complete.

**Return Format** <value><NL>  
<value> ::= integer in NR1 format.

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[:CHANnel<n>:PROtection](#)" on page 295
- "[:OPEE \(Operation Status Enable Register\)](#)" on page 225
- "[:OPERregister:CONDition \(Operation Status Condition Register\)](#)" on page 227
- "[:OVLenable \(Overload Event Enable Register\)](#)" on page 231

- [":OVLRegister \(Overload Event Register\)" on page 233](#)
- ["\\*STB \(Read Status Byte\)" on page 198](#)
- ["\\*SRE \(Service Request Enable\)" on page 196](#)

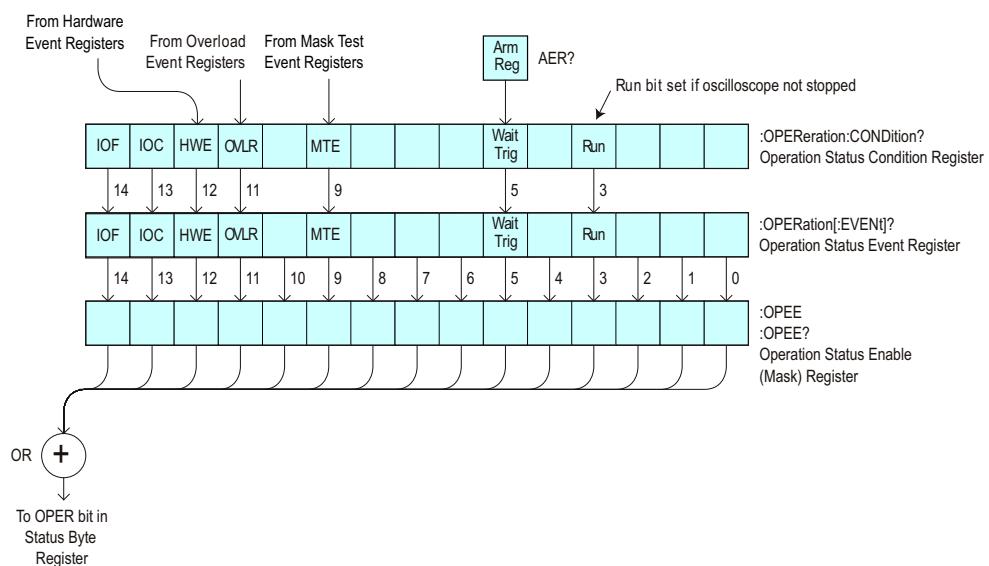
## :OPEE (Operation Status Enable Register)

 (see page 1334)

**Command Syntax** :OPEE <mask>

<mask> ::= 15-bit integer

The :OPEE command sets a mask in the Operation Status Enable register. Set any of the following bits to "1" to enable bit 7 in the Status Byte Register and potentially cause an SRQ (Service Request) interrupt to be generated.



**Table 83** Operation Status Enable Register (OPEE)

Bit	Name	Description	When Set (1 = High = True), Enables:
14	IOF	IO Operation Failed	Event when the IO operation fails.
13	IOC	IO Operation Complete	Event when the IO operation completes.
12	HWE	Hardware Event	Event when hardware event occurs.
11	OVLR	Overload	Event when 50Ω input overload occurs.
10	---	---	(Not used.)
9	MTE	Mask Test Event	Event when mask test event occurs.
8-6	---	---	(Not used.)
5	Wait Trig	Wait Trig	Event when the trigger is armed.
4	---	---	(Not used.)

**Table 83** Operation Status Enable Register (OPEE) (continued)

Bit	Name	Description	When Set (1 = High = True), Enables:
3	Run	Running	Event when the oscilloscope is running (not stopped).
2-0	---	---	(Not used.)

Query Syntax :OPEE?

The :OPEE? query returns the current value contained in the Operation Status Enable register as an integer number.

Return Format &lt;value&gt;&lt;NL&gt;

```
<value> ::= integer in NR1 format.
```

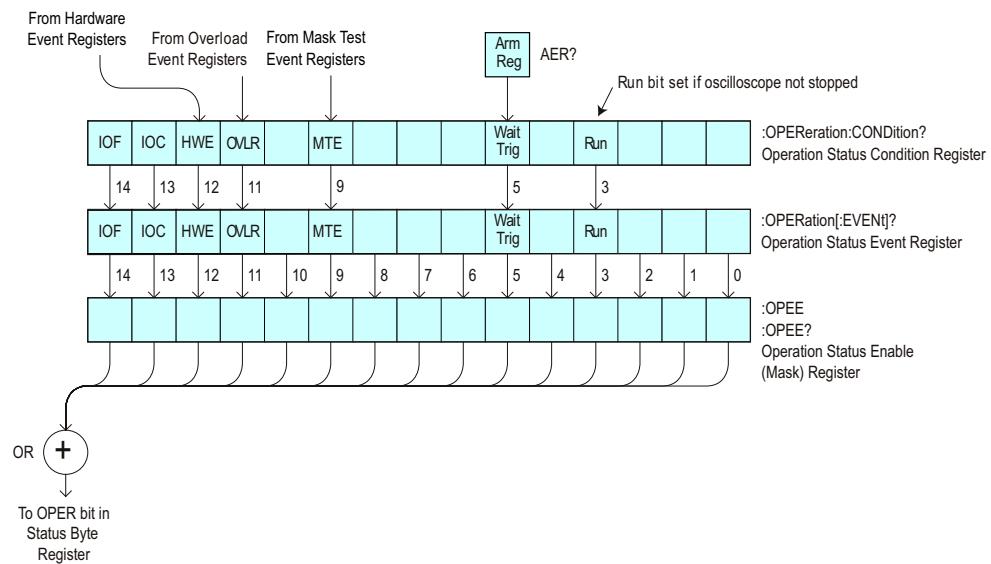
- See Also
- "[Introduction to Root \(:\) Commands](#)" on page 206
  - "[":AER \(Arm Event Register\)](#)" on page 208
  - "[":CHANnel<n>:PROTection](#)" on page 295
  - "[":OPERegister\[:EVENT\]](#) ([Operation Status Event Register](#))" on page 229
  - "[":OVLenable \(Overload Event Enable Register\)"](#) on page 231
  - "[":OVLRegister \(Overload Event Register\)"](#) on page 233
  - "[":\\*STB \(Read Status Byte\)"](#) on page 198
  - "[":\\*SRE \(Service Request Enable\)"](#) on page 196

## :OPERegister:CONDition (Operation Status Condition Register)

**C** (see page 1334)

**Query Syntax** :OPERegister:CONDition?

The :OPERegister:CONDition? query returns the integer value contained in the Operation Status Condition Register.



**Table 84** Operation Status Condition Register

Bit	Name	Description	When Set (1 = High = True), Indicates:
14	IOF	IO Operation Failed	Event when the IO operation fails.
13	IOC	IO Operation Complete	Event when the IO operation completes.
12	HWE	Hardware Event	Event when hardware event occurs.
11	OVLR	Overload	A 50Ω input overload has occurred.
10	---	---	(Not used.)
9	MTE	Mask Test Event	A mask test event has occurred.
8-6	---	---	(Not used.)
5	Wait Trig	Wait Trig	The trigger is armed (set by the Trigger Armed Event Register (TER)).
4	---	---	(Not used.)
3	Run	Running	The oscilloscope is running (not stopped).
2-0	---	---	(Not used.)

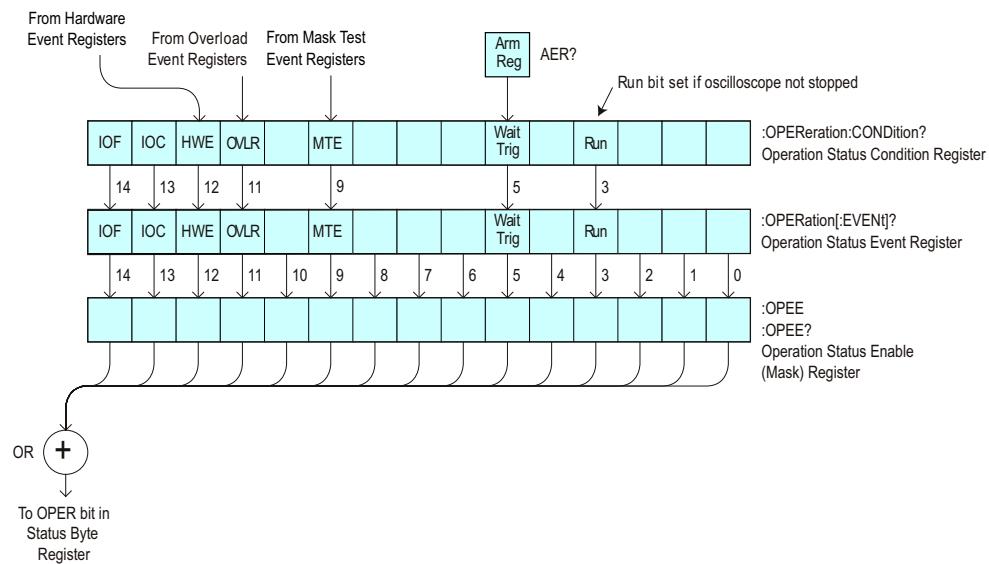
Return Format	<value><NL>  <value> ::= integer in NR1 format.
See Also	<ul style="list-style-type: none"><li>· "<a href="#">Introduction to Root (:) Commands</a>" on page 206</li><li>· "<a href="#">:CHANnel&lt;n&gt;:PROTection</a>" on page 295</li><li>· "<a href="#">:OPEE (Operation Status Enable Register)</a>" on page 225</li><li>· "<a href="#">:OPERegister[:EVENT] (Operation Status Event Register)</a>" on page 229</li><li>· "<a href="#">:OVLenable (Overload Event Enable Register)</a>" on page 231</li><li>· "<a href="#">:OVLRegister (Overload Event Register)</a>" on page 233</li><li>· "<a href="#">:*STB (Read Status Byte)</a>" on page 198</li><li>· "<a href="#">:*SRE (Service Request Enable)</a>" on page 196</li><li>· "<a href="#">:MTERegister[:EVENT] (Mask Test Event Event Register)</a>" on page 223</li><li>· "<a href="#">:MTEenable (Mask Test Event Enable Register)</a>" on page 221</li></ul>

## :OPERegister[:EVENT] (Operation Status Event Register)

**C** (see page 1334)

**Query Syntax** :OPERegister [:EVENT] ?

The :OPERegister[:EVENT]? query returns the integer value contained in the Operation Status Event Register.



**Table 85** Operation Status Event Register

Bit	Name	Description	When Set (1 = High = True), Indicates:
14	IOF	IO Operation Failed	Event when the IO operation fails.
13	IOC	IO Operation Complete	Event when the IO operation completes.
12	HWE	Hardware Event	Event when hardware event occurs.
11	OVLR	Overload	A 50Ω input overload has occurred.
10	---	---	(Not used.)
9	MTE	Mask Test Event	A mask test event has occurred.
8-6	---	---	(Not used.)
5	Wait Trig	Wait Trig	The trigger is armed (set by the Trigger Armed Event Register (TER)).
4	---	---	(Not used.)
3	Run	Running	The oscilloscope has gone from a stop state to a single or running state.
2-0	---	---	(Not used.)

Return Format	<value><NL>  <value> ::= integer in NR1 format.
See Also	<ul style="list-style-type: none"><li>· "<a href="#">Introduction to Root (:) Commands</a>" on page 206</li><li>· "<a href="#">:CHANnel&lt;n&gt;:PROTection</a>" on page 295</li><li>· "<a href="#">:OPEE (Operation Status Enable Register)</a>" on page 225</li><li>· "<a href="#">:OPERegister:CONDition (Operation Status Condition Register)</a>" on page 227</li><li>· "<a href="#">:OVLenable (Overload Event Enable Register)</a>" on page 231</li><li>· "<a href="#">:OVLRegister (Overload Event Register)</a>" on page 233</li><li>· "<a href="#">:*STB (Read Status Byte)</a>" on page 198</li><li>· "<a href="#">:*SRE (Service Request Enable)</a>" on page 196</li><li>· "<a href="#">:MTERegister[:EVENT] (Mask Test Event Event Register)</a>" on page 223</li><li>· "<a href="#">:MTEenable (Mask Test Event Enable Register)</a>" on page 221</li></ul>

## :OVLenable (Overload Event Enable Register)

**C** (see [page 1334](#))

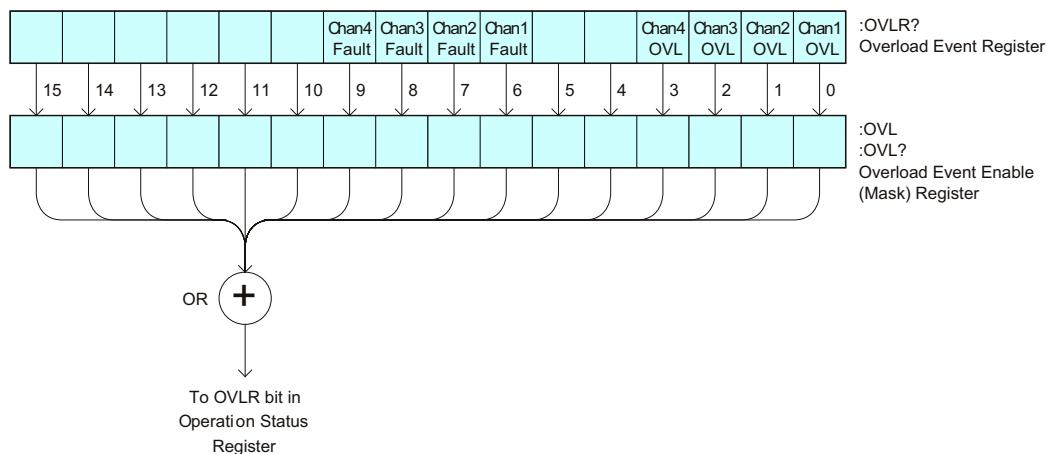
**Command Syntax** :OVLenable <enable\_mask>  
 <enable\_mask> ::= 16-bit integer

The overload enable mask is an integer representing an input as described in the following table.

The :OVLenable command sets the mask in the Overload Event Enable Register and enables the reporting of the Overload Event Register. If an overvoltage is sensed on a  $50\Omega$  input, the input will automatically switch to  $1 M\Omega$  input impedance. If enabled, such an event will set bit 11 in the Operation Status Register.

### NOTE

You can set analog channel input impedance to  $50\Omega$  on the 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models. On these same bandwidth models, if there are only two analog channels, you can also set external trigger input impedance to  $50\Omega$ .



**Table 86** Overload Event Enable Register (OVL)

Bit	Description	When Set (1 = High = True), Enables:
15-10	---	(Not used.)
9	Channel 4 Fault	Event when fault occurs on Channel 4 input.
8	Channel 3 Fault	Event when fault occurs on Channel 3 input.
7	Channel 2 Fault	Event when fault occurs on Channel 2 input.
6	Channel 1 Fault	Event when fault occurs on Channel 1 input.
5-4	---	(Not used.)

**Table 86** Overload Event Enable Register (OVL) (continued)

Bit	Description	When Set (1 = High = True), Enables:
3	Channel 4 OVL	Event when overload occurs on Channel 4 input.
2	Channel 3 OVL	Event when overload occurs on Channel 3 input.
1	Channel 2 OVL	Event when overload occurs on Channel 2 input.
0	Channel 1 OVL	Event when overload occurs on Channel 1 input.

**Query Syntax** :OVLenable?

The :OVLenable query returns the current enable mask value contained in the Overload Event Enable Register.

**Return Format**

```
<enable_mask><NL>
<enable_mask> ::= integer in NR1 format.
```

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":CHANnel<n>:PROTection](#)" on page 295
- "[":OPEE \(Operation Status Enable Register\)](#)" on page 225
- "[":OPERegister:CONDition \(Operation Status Condition Register\)"](#) on page 227
- "[":OPERegister\[:EVENT\] \(Operation Status Event Register\)"](#) on page 229
- "[":OVLRegister \(Overload Event Register\)"](#) on page 233
- "[":\\*STB \(Read Status Byte\)"](#) on page 198
- "[":\\*SRE \(Service Request Enable\)"](#) on page 196

## :OVLRegister (Overload Event Register)

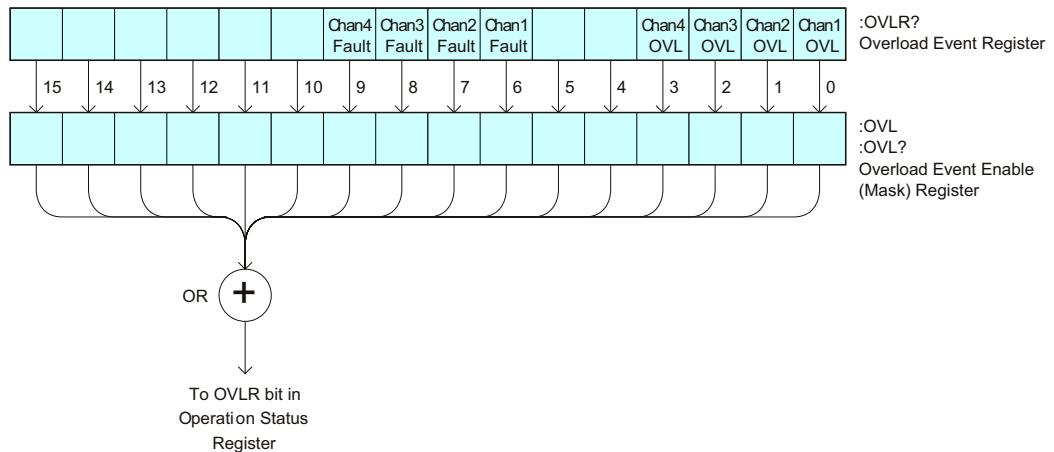
**C** (see page 1334)

**Query Syntax** :OVLRegister?

The :OVLRegister query returns the overload protection value stored in the Overload Event Register (OCLR). If an overvoltage is sensed on a  $50\Omega$  input, the input will automatically switch to  $1 M\Omega$  input impedance. A "1" indicates an overload has occurred.

**NOTE**

You can set analog channel input impedance to  $50\Omega$  on the 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models. On these same bandwidth models, if there are only two analog channels, you can also set external trigger input impedance to  $50\Omega$ .



**Table 87** Overload Event Register (OCLR)

Bit	Description	When Set (1 = High = True), Indicates:
15-10	---	(Not used.)
9	Channel 4 Fault	Fault has occurred on Channel 4 input.
8	Channel 3 Fault	Fault has occurred on Channel 3 input.
7	Channel 2 Fault	Fault has occurred on Channel 2 input.
6	Channel 1 Fault	Fault has occurred on Channel 1 input.
5-4	---	(Not used.)
3	Channel 4 OVL	Overload has occurred on Channel 4 input.
2	Channel 3 OVL	Overload has occurred on Channel 3 input.

**Table 87** Overload Event Register (OVLR) (continued)

Bit	Description	When Set (1 = High = True), Indicates:
1	Channel 2 OVL	Overload has occurred on Channel 2 input.
0	Channel 1 OVL	Overload has occurred on Channel 1 input.

**Return Format** <value><NL>  
 <value> ::= integer in NR1 format.

- See Also**
- "[Introduction to Root \(:\) Commands](#)" on page 206
  - "[:CHANnel<n>:PROTection](#)" on page 295
  - "[:OPEE \(Operation Status Enable Register\)](#)" on page 225
  - "[:OVLenable \(Overload Event Enable Register\)](#)" on page 231
  - "[\\*STB \(Read Status Byte\)](#)" on page 198
  - "[\\*SRE \(Service Request Enable\)](#)" on page 196

**:PRINT****C** (see [page 1334](#))

<b>Command Syntax</b>	<code>:PRINT [&lt;options&gt;]</code>  <code>&lt;options&gt; ::= [&lt;print option&gt;] [,...,&lt;print option&gt;]</code>  <code>&lt;print option&gt; ::= {COLOR   GRAYscale   PRINTER0   PRINTER1   BMP8bit   BMP   PNG   NOFactors   FACTors}</code>
	The <code>&lt;print option&gt;</code> parameter may be repeated up to 5 times.
	The PRINT command formats the output according to the currently selected format (device). If an option is not specified, the value selected in the Print Config menu is used.

**See Also**

- "[Introduction to Root \(\) Commands](#)" on page 206
- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[:HARDcopy:FACTors](#)" on page 421
- "[:HARDcopy:GRAYscale](#)" on page 1257
- "[:DISPLAY:DATA](#)" on page 339

**:RUN****C** (see [page 1334](#))**Command Syntax** :RUN

The :RUN command starts repetitive acquisitions. This is the same as pressing the Run key on the front panel.

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":SINGle"](#) on page 238
- "[":STOP"](#) on page 240

**Example Code**

```
' RUN_STOP - (not executed in this example)
'   - RUN starts the data acquisition for the active waveform display.
'   - STOP stops the data acquisition and turns off AUTOSTORE.
' myScope.WriteString ":RUN"      ' Start data acquisition.
' myScope.WriteString ":STOP"     ' Stop the data acquisition.
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on [page 1343](#)

**:SERial**

**N** (see [page 1334](#))

**Query Syntax** :SERial?

The :SERial? query returns the serial number of the instrument.

**Return Format:** Unquoted string<NL>

**See Also** • ["Introduction to Root \(:\) Commands" on page 206](#)

## :SINGle

 (see [page 1334](#))

### Command Syntax

`:SINGle`

The :SINGle command causes the instrument to acquire a single trigger of data. This is the same as pressing the Single key on the front panel.

### See Also

- ["Introduction to Root \(:\) Commands" on page 206](#)
- [":RUN" on page 236](#)
- [":STOP" on page 240](#)

**:STATUs**

**N** (see [page 1334](#))

**Query Syntax**

```
:STATUs? <source>

<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for the DSO models

<source> ::= {CHANnel<n> | DIGital<d> | POD{1 | 2}
              | BUS{1 | 2} | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for the MSO models

<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The .STATUs? query reports whether the channel, function, serial decode bus, or reference waveform location specified by <source> is displayed.

**NOTE**

MATH<m> is an alias for FUNCtion<m>.

**Return Format**

```
<value><NL>
<value> ::= {1 | 0}
```

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[:BLANK](#)" on page 214
- "[:CHANnel<n>:DISPlay](#)" on page 284
- "[:DIGital<d>:DISPlay](#)" on page 323
- "[:FUNCtion<m>:DISPlay](#)" on page 391
- "[:POD<n>:DISPlay](#)" on page 603
- "[:SBUS<n>:DISPlay](#)" on page 726
- "[:WMEMory<r>:DISPlay](#)" on page 1226
- "[:VIEW](#)" on page 242

## :STOP

 (see [page 1334](#))

**Command Syntax** :STOP

The :STOP command stops the acquisition. This is the same as pressing the Stop key on the front panel.

**See Also**

- "[Introduction to Root \(:\) Commands](#)" on page 206
- "[":RUN"](#) on page 236
- "[":SINGle"](#) on page 238

**Example Code**

- "[":Example Code"](#) on page 236

## :TER (Trigger Event Register)



(see [page 1334](#))

**Query Syntax** :TER?

The :TER? query reads the Trigger Event Register. After the Trigger Event Register is read, it is cleared. A one indicates a trigger has occurred. A zero indicates a trigger has not occurred.

The Trigger Event Register is summarized in the TRG bit of the Status Byte Register (STB). A Service Request (SRQ) can be generated when the TRG bit of the Status Byte transitions, and the TRG bit is set in the Service Request Enable register. The Trigger Event Register must be cleared each time you want a new service request to be generated.

**Return Format** <value><NL>

<value> ::= {1 | 0}; a 16-bit integer in NR1 format.

**See Also**

- ["Introduction to Root \(:\) Commands"](#) on page 206
- ["\\*SRE \(Service Request Enable\)"](#) on page 196
- ["\\*STB \(Read Status Byte\)"](#) on page 198

**:VIEW**

**N** (see [page 1334](#))

**Command Syntax**

```
:VIEW <source>

<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for DSO models

<source> ::= {CHANnel<n> | DIGItal<d> | POD{1 | 2}
              | BUS{1 | 2} | FUNCtion<m> | MATH<m> | FFT | SBUS{1 | 2}
              | WMEMory<r>}
              for MSO models

<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :VIEW command turns on the specified channel, function, serial decode bus, or reference waveform location.

**NOTE**

MATH<m> is an alias for FUNCtion<m>.

**See Also**

- ["Introduction to Root \(:\) Commands" on page 206](#)
- [":BLANK" on page 214](#)
- [":CHANnel<n>:DISPlay" on page 284](#)
- [":DIGItal<d>:DISPlay" on page 323](#)
- [":FUNCtion<m>:DISPlay" on page 391](#)
- [":POD<n>:DISPlay" on page 603](#)
- [":SBUS<n>:DISPlay" on page 726](#)
- [":WMEMory<r>:DISPlay" on page 1226](#)
- [":STATus" on page 239](#)

**Example Code**

```
' VIEW_BLANK - (not executed in this example)
'   - VIEW turns on (starts displaying) a channel.
'   - BLANK turns off (stops displaying) a channel.
' myScope.WriteString ":BLANK CHANnel1"    ' Turn channel 1 off.
' myScope.WriteString ":VIEW CHANnel1"     ' Turn channel 1 on.
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

# 7 :ACQuire Commands

Set the parameters for acquiring and storing data. See "[Introduction to :ACQuire Commands](#)" on page 243.

**Table 88** :ACQuire Commands Summary

Command	Query	Options and Query Returns
:ACQuire:COMPLETE <complete> (see page 245)	:ACQuire:COMPLETE? (see <a href="#">page 245</a> )	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNT <count> (see page 246)	:ACQuire:COUNT? (see <a href="#">page 246</a> )	<count> ::= an integer from 2 to 65536 in NR1 format
:ACQuire:MODE <mode> (see page 247)	:ACQuire:MODE? (see <a href="#">page 247</a> )	<mode> ::= {RTIMe   ETIMe   SEGmented}
n/a	:ACQuire:POINTS? (see <a href="#">page 248</a> )	<# points> ::= an integer in NR1 format
:ACQuire:SEGMENTed:ANALyze (see page 249)	n/a	n/a (with Option SGM)
:ACQuire:SEGMENTed:COUNT <count> (see page 250)	:ACQuire:SEGMENTed:COUNT? (see <a href="#">page 250</a> )	<count> ::= an integer from 2 to 1000 in NR1 format (with Option SGM)
:ACQuire:SEGMENTed:INDEX <index> (see page 251)	:ACQuire:SEGMENTed:INDEX? (see <a href="#">page 251</a> )	<index> ::= an integer from 1 to 1000 in NR1 format (with Option SGM)
n/a	:ACQuire:SRATE? (see <a href="#">page 254</a> )	<sample_rate> ::= sample rate (samples/s) in NR3 format
:ACQuire:TYPE <type> (see page 255)	:ACQuire:TYPE? (see <a href="#">page 255</a> )	<type> ::= {NORMAL   AVERage   HRESolution   PEAK}

**Introduction to :ACQuire Commands** The ACQuire subsystem controls the way in which waveforms are acquired. These acquisition types are available: normal, averaging, peak detect, and high resolution.

Normal

The :ACQuire:TYPE NORMal command sets the oscilloscope in the normal acquisition mode. For the majority of user models and signals, NORMal mode yields the best oscilloscope picture of the waveform.

### Averaging

The :ACQuire:TYPE AVERage command sets the oscilloscope in the averaging mode. You can set the count by sending the :ACQuire:COUNt command followed by the number of averages. In this mode, the value for averages is an integer from 2 to 65536. The COUNt value determines the number of averages that must be acquired.

### High-Resolution

The :ACQuire:TYPE HRESolution command sets the oscilloscope in the high-resolution mode (also known as *smoothing*). This mode is used to reduce noise at slower sweep speeds where the digitizer samples faster than needed to fill memory for the displayed time range. Instead of decimating samples, they are averaged together to provide the value for one display point. The slower the sweep speed, the greater the number of samples that are averaged together for each display point.

### Peak Detect

The :ACQuire:TYPE PEAK command sets the oscilloscope in the peak detect mode. In this mode, :ACQuire:COUNt has no meaning.

### Reporting the Setup

Use :ACQuire? to query setup information for the ACQuire subsystem.

### Return Format

The following is a sample response from the :ACQuire? query. In this case, the query was issued following a \*RST command.

```
:ACQ:MODE RTIM;TYPE NORM;COMP 100;COUNT 8;SEGM:COUN 2
```

## :ACQuire:COMplete

**C** (see [page 1334](#))

**Command Syntax** :ACQuire:COMplete <complete>

<complete> ::= 100; an integer in NR1 format

The :ACQuire:COMplete command affects the operation of the :DIGitize command. It specifies the minimum completion criteria for an acquisition. The parameter determines the percentage of the time buckets that must be "full" before an acquisition is considered complete. If :ACQuire:TYPE is NORMAl, it needs only one sample per time bucket for that time bucket to be considered full.

The only legal value for the :COMplete command is 100. All time buckets must contain data for the acquisition to be considered complete.

**Query Syntax** :ACQuire:COMplete?

The :ACQuire:COMplete? query returns the completion criteria (100) for the currently selected mode.

**Return Format** <completion\_criteria><NL>

<completion\_criteria> ::= 100; an integer in NR1 format

- See Also**
- "[Introduction to :ACQuire Commands](#)" on page 243
  - "[":ACQuire:TYPE](#)" on page 255
  - "[":DIGITIZE](#)" on page 215
  - "[":WAVEFORM:POINTS](#)" on page 1158

**Example Code**

```
' AQUIRE_COMPLETE - Specifies the minimum completion criteria for
' an acquisition. The parameter determines the percentage of time
' buckets needed to be "full" before an acquisition is considered
' to be complete.
myScope.WriteString ":ACQuire:COMplete 100"
```

See complete example programs at: [Chapter 42](#), "Programming Examples," starting on page 1343

## :ACQuire:COUNt



(see [page 1334](#))

### Command Syntax

```
:ACQuire:COUNt <count>
<count> ::= integer in NR1 format
```

In averaging mode, the :ACQuire:COUNt command specifies the number of values to be averaged for each time bucket before the acquisition is considered to be complete for that time bucket. When :ACQuire:TYPE is set to AVERage, the count can be set to any value from 2 to 65536.

### NOTE

The :ACQuire:COUNt 1 command has been deprecated. The AVERage acquisition type with a count of 1 is functionally equivalent to the HRESolution acquisition type; however, you should select the high-resolution acquisition mode with the :ACQuire:TYPE HRESolution command instead.

### Query Syntax

```
:ACQuire:COUNT?
```

The :ACQuire:COUNT? query returns the currently selected count value for averaging mode.

### Return Format

```
<count_argument><NL>
<count_argument> ::= an integer from 2 to 65536 in NR1 format
```

### See Also

- ["Introduction to :ACQuire Commands"](#) on page 243
- [":ACQuire:TYPE"](#) on page 255
- [":DIGitize"](#) on page 215
- [":WAVeform:COUNt"](#) on page 1154

## :ACQuire:MODE

**C** (see [page 1334](#))

**Command Syntax** :ACQuire:MODE <mode>

<mode> ::= {RTIMe | SEGmented}

The :ACQuire:MODE command sets the acquisition mode of the oscilloscope.

- The :ACQuire:MODE RTIMe command sets the oscilloscope in real time mode.

### NOTE

The obsolete command ACQuire:TYPE:REALtime is functionally equivalent to sending ACQuire:MODE RTIMe; TYPE NORMAl.

- The :ACQuire:MODE SEGmented command sets the oscilloscope in segmented memory mode.

**Query Syntax** :ACQuire:MODE?

The :ACQuire:MODE? query returns the acquisition mode of the oscilloscope.

**Return Format** <mode\_argument><NL>

<mode\_argument> ::= {RTIM | SEGM}

- See Also**
- "[Introduction to :ACQuire Commands](#)" on page 243
  - "[":ACQuire:TYPE"](#) on page 255

**:ACQuire:POINts****C** (see [page 1334](#))**Query Syntax** `:ACQuire:POINts?`

The `:ACQuire:POINts?` query returns the number of data points that the hardware will acquire from the input signal. The number of points acquired is not directly controllable. To set the number of points to be transferred from the oscilloscope, use the command `:WAVeform:POINts`. The `:WAVeform:POINts?` query will return the number of points available to be transferred from the oscilloscope.

**Return Format** `<points_argument><NL>`

`<points_argument>` ::= an integer in NR1 format

**See Also** • ["Introduction to :ACQuire Commands"](#) on page 243

- [":DIGitize"](#) on page 215

- [":WAVeform:POINts"](#) on page 1158

## :ACQuire:SEGmented:ANALyze

**N** (see [page 1334](#))

**Command Syntax** :ACQuire:SEGmented:ANALyze

### NOTE

This command is available when the segmented memory option (Option SGM) is enabled.

---

This command calculates measurement statistics and/or infinite persistence over all segments that have been acquired. It corresponds to the front panel **Analyze Segments** softkey which appears in both the Measurement Statistics and Segmented Memory Menus.

In order to use this command, the oscilloscope must be stopped and in segmented acquisition mode, with either quick measurements or infinite persistence on.

**See Also**

- [":ACQuire:MODE"](#) on page 247
- [":ACQuire:SEGmented:COUNt"](#) on page 250
- ["Introduction to :ACQuire Commands"](#) on page 243

## :ACQuire:SEGmented:COUNt

**N** (see [page 1334](#))

**Command Syntax**    `:ACQuire:SEGmented:COUNt <count>`  
`<count> ::= an integer from 2 to 1000 (w/4M memory) in NR1 format`

### NOTE

This command is available when the segmented memory option (Option SGM) is enabled.

The :ACQuire:SEGmented:COUNt command sets the number of memory segments to acquire.

The segmented memory acquisition mode is enabled with the :ACQuire:MODE command, and data is acquired using the :DIGitize, :SINGle, or :RUN commands. The number of memory segments in the current acquisition is returned by the :WAVeform:SEGmented:COUNt? query.

The maximum number of segments may be limited by the memory depth of your oscilloscope. For example, an oscilloscope with 1M memory allows a maximum of 250 segments.

**Query Syntax**    `:ACQuire:SEGmented:COUNt?`

The :ACQuire:SEGmented:COUNt? query returns the current count setting.

**Return Format**    `<count><NL>`  
`<count> ::= an integer from 2 to 1000 (w/4M memory) in NR1 format`

**See Also**

- "[:ACQuire:MODE](#)" on page 247
- "[:DIGITIZE](#)" on page 215
- "[:SINGLE](#)" on page 238
- "[:RUN](#)" on page 236
- "[:WAVeform:SEGmented:COUNt](#)" on page 1165
- "[:ACQuire:SEGmented:ANALyze](#)" on page 249
- "[Introduction to :ACQuire Commands](#)" on page 243

**Example Code**

- "["Example Code"](#) on page 251

## :ACQuire:SEGmented:INDEX

**N** (see [page 1334](#))

**Command Syntax** :ACQuire:SEGmented:INDEX <index>  
 <index> ::= an integer from 1 to 1000 (w/4M memory) in NR1 format

### NOTE

This command is available when the segmented memory option (Option SGM) is enabled.

The :ACQuire:SEGmented:INDEX command sets the index into the memory segments that have been acquired.

The segmented memory acquisition mode is enabled with the :ACQuire:MODE command. The number of segments to acquire is set using the :ACQuire:SEGmented:COUNt command, and data is acquired using the :DIGitize, :SINGle, or :RUN commands. The number of memory segments that have been acquired is returned by the :WAVeform:SEGmented:COUNt? query. The time tag of the currently indexed memory segment is returned by the :WAVeform:SEGmented:TTAG? query.

The maximum number of segments may be limited by the memory depth of your oscilloscope. For example, an oscilloscope with 1M memory allows a maximum of 250 segments.

**Query Syntax** :ACQuire:SEGmented:INDEX?

The :ACQuire:SEGmented:INDEX? query returns the current segmented memory index setting.

**Return Format** <index><NL>  
 <index> ::= an integer from 1 to 1000 (w/4M memory) in NR1 format

**See Also**

- [":ACQuire:MODE"](#) on page 247
- [":ACQuire:SEGmented:COUNt"](#) on page 250
- [":DIGitize"](#) on page 215
- [":SINGle"](#) on page 238
- [":RUN"](#) on page 236
- [":WAVeform:SEGmented:COUNt"](#) on page 1165
- [":WAVeform:SEGmented:TTAG"](#) on page 1166
- [":ACQuire:SEGmented:ANALyze"](#) on page 249
- ["Introduction to :ACQuire Commands"](#) on page 243

**Example Code**

```
' Segmented memory commands example.  

' -----
```

```

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub Main()

On Error GoTo VisaComError

' Create the VISA COM I/O resource.
Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488
Set myScope.IO =
    myMgr.Open("USB0::0x0957::0x17A6::US50210029::0::INSTR")
myScope.IO.Clear      ' Clear the interface.

' Turn on segmented memory acquisition mode.
myScope.WriteString ":ACQuire:MODE SEGmented"
myScope.WriteString ":ACQuire:MODE?"
strQueryResult = myScope.ReadString
Debug.Print "Acquisition mode: " + strQueryResult

' Set the number of segments to 25.
myScope.WriteString ":ACQuire:SEGmented:COUNT 25"
myScope.WriteString ":ACQuire:SEGmented:COUNT?"
strQueryResult = myScope.ReadString
Debug.Print "Acquisition memory segments: " + strQueryResult

' If data will be acquired within the IO timeout:
'myScope.IO.Timeout = 10000
'myScope.WriteString ":DIGitize"
'Debug.Print ":DIGITIZE blocks until all segments acquired."
'myScope.WriteString ":WAVEform:SEGmented:COUNT?"
'varQueryResult = myScope.ReadNumber

' Or, to poll until the desired number of segments acquired:
myScope.WriteString ":SINGLE"
Debug.Print ":SINGLE does not block until all segments acquired."
Do
    Sleep 100      ' Small wait to prevent excessive queries.
    myScope.WriteString ":WAVEform:SEGmented:COUNT?"
    varQueryResult = myScope.ReadNumber
Loop Until varQueryResult = 25

Debug.Print "Number of segments in acquired data: " _
    + FormatNumber(varQueryResult)

Dim lngSegments As Long
lngSegments = varQueryResult

' For each segment:
Dim dblTimeTag As Double
Dim lngI As Long

```

```
For lngI = lngSegments To 1 Step -1

    ' Set the segmented memory index.
    myScope.WriteString ":ACQuire:SEGmented:INDEX " + CStr(lngI)
    myScope.WriteString ":ACQuire:SEGmented:INDEX?"
    strQueryResult = myScope.ReadString
    Debug.Print "Acquisition memory segment index: " + strQueryResult

    ' Display the segment time tag.
    myScope.WriteString ":WAVEform:SEGmented:TTAG?"
    dblTimeTag = myScope.ReadNumber
    Debug.Print "Segment " + CStr(lngI) + " time tag: " _
        + FormatNumber(dblTimeTag, 12)

    Next lngI

    Exit Sub

VisaComError:
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub
```

**:ACQuire:SRATe**

**N** (see [page 1334](#))

**Query Syntax**    `:ACQuire:SRATe? [MAXimum]`

The `:ACQuire:SRATe?` query returns the current oscilloscope acquisition sample rate. The sample rate is not directly controllable.

When the MAXimum parameter is used, the oscilloscope's maximum possible sample rate is returned.

**Return Format**    `<sample_rate><NL>`

`<sample_rate>` ::= sample rate in NR3 format

**See Also**

- "Introduction to [:ACQuire Commands](#)" on page 243
- "[:ACQuire:POINTs](#)" on page 248

## :ACQuire:TYPE

**C** (see [page 1334](#))

**Command Syntax** :ACQuire:TYPE <type>

<type> ::= {NORMAL | AVERage | HRESolution | PEAK}

The :ACQuire:TYPE command selects the type of data acquisition that is to take place. The acquisition types are:

- NORMAL – sets the oscilloscope in the normal mode.
- AVERage – sets the oscilloscope in the averaging mode. You can set the count by sending the :ACQuire:COUNt command followed by the number of averages. In this mode, the value for averages is an integer from 1 to 65536. The COUNt value determines the number of averages that must be acquired.

The AVERage type is not available when in segmented memory mode (:ACQuire:MODE SEGmented).

- HRESolution – sets the oscilloscope in the high-resolution mode (also known as *smoothing*). This mode is used to reduce noise at slower sweep speeds where the digitizer samples faster than needed to fill memory for the displayed time range.

For example, if the digitizer samples at 200 MSa/s, but the effective sample rate is 1 MSa/s (because of a slower sweep speed), only 1 out of every 200 samples needs to be stored. Instead of storing one sample (and throwing others away), the 200 samples are averaged together to provide the value for one display point. The slower the sweep speed, the greater the number of samples that are averaged together for each display point.

- PEAK – sets the oscilloscope in the peak detect mode. In this mode, :ACQuire:COUNt has no meaning.

The AVERage and HRESolution types can give you extra bits of vertical resolution. See the *User's Guide* for an explanation. When getting waveform data acquired using the AVERage and HRESolution types, be sure to use the WORD or ASCII waveform data formats to get the extra bits of vertical resolution.

### NOTE

The obsolete command ACQuire:TYPE:REALtime is functionally equivalent to sending ACQuire:MODE RTIME; TYPE NORMAL.

**Query Syntax** :ACQuire:TYPE?

The :ACQuire:TYPE? query returns the current acquisition type.

**Return Format** <acq\_type><NL>

<acq\_type> ::= {NORM | AVER | HRES | PEAK}

**See Also** • "[Introduction to :ACQuire Commands](#)" on page 243

- "[:ACQuire:COUNT](#)" on page 246
- "[:ACQuire:MODE](#)" on page 247
- "[:DIGitize](#)" on page 215
- "[:WAVeform:FORMat](#)" on page 1157
- "[:WAVeform:TYPE](#)" on page 1172
- "[:WAVeform:PREamble](#)" on page 1162

**Example Code**

```
' AQUIRE_TYPE - Sets the acquisition mode, which can be NORMAL,  
' PEAK, or AVERAGE.  
myScope.WriteString ":ACQuire:TYPE NORMAL"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## 8 :BUS<n> Commands

Control all oscilloscope functions associated with buses made up of digital channels. See "[Introduction to :BUS<n> Commands](#)" on page 258.

**Table 89** :BUS<n> Commands Summary

Command	Query	Options and Query Returns
:BUS<n>:BIT<m> {{0   OFF}   {1   ON}} (see <a href="#">page 259</a> )	:BUS<n>:BIT<m>? (see <a href="#">page 259</a> )	{0   1} <n> ::= 1 or 2; an integer in NR1 format <m> ::= 0-15; an integer in NR1 format
:BUS<n>:BITS <channel_list>, {{0   OFF}   {1   ON}} (see <a href="#">page 260</a> )	:BUS<n>:BITS? (see <a href="#">page 260</a> )	<channel_list>, {0   1} <channel_list> ::= (@<m>,<m>:<m> ... ) where "," is separator and ":" is range <n> ::= 1 or 2; an integer in NR1 format <m> ::= 0-15; an integer in NR1 format
:BUS<n>:CLEar (see <a href="#">page 262</a> )	n/a	<n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:DISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 263</a> )	:BUS<n>:DISPlay? (see <a href="#">page 263</a> )	{0   1} <n> ::= 1 or 2; an integer in NR1 format

**Table 89** :BUS<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:BUS<n>:LABEL <string> (see page 264)	:BUS<n>:LABEL? (see page 264)	<string> ::= quoted ASCII string up to 10 characters <n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:MASK <mask> (see page 265)	:BUS<n>:MASK? (see page 265)	<mask> ::= 32-bit integer in decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal <n> ::= 1 or 2; an integer in NR1 format

**Introduction to  
:BUS<n>  
Commands**

<n> ::= {1 | 2}

The BUS subsystem commands control the viewing, labeling, and digital channel makeup of two possible buses.

#### NOTE

These commands are only valid for the MSO models.

#### Reporting the Setup

Use :BUS<n>? to query setup information for the BUS subsystem.

#### Return Format

The following is a sample response from the :BUS1? query. In this case, the query was issued following a \*RST command.

```
:BUS1:DISP 0;LAB "BUS1";MASK +255
```

## :BUS<n>:BIT<m>

**N** (see [page 1334](#))

### Command Syntax

```
:BUS<n>:BIT<m> <display>
<display> ::= {{1 | ON} | {0 | OFF}}
<n> ::= An integer, 1 or 2, is attached as a suffix to BUS
and defines the bus that is affected by the command.
<m> ::= An integer, 0,...,15, is attached as a suffix to BIT
and defines the digital channel that is affected by the command.
```

The :BUS<n>:BIT<m> command includes or excludes the selected bit as part of the definition for the selected bus. If the parameter is a 1 (ON), the bit is included in the definition. If the parameter is a 0 (OFF), the bit is excluded from the definition. Note: BIT0-15 correspond to DIGital0-15.

### NOTE

This command is only valid for the MSO models.

### Query Syntax

```
:BUS<n>:BIT<m>?
```

The :BUS<n>:BIT<m>? query returns the value indicating whether the specified bit is included or excluded from the specified bus definition.

### Return Format

```
<display><NL>
<display> ::= {0 | 1}
```

### See Also

- "[Introduction to :BUS<n> Commands](#)" on page 258
- "[":BUS<n>:BITS](#)" on page 260
- "[":BUS<n>:CLEar](#)" on page 262
- "[":BUS<n>:DISPlay](#)" on page 263
- "[":BUS<n>:LABEL](#)" on page 264
- "[":BUS<n>:MASK](#)" on page 265

### Example Code

```
' Include digital channel 1 in bus 1:
myScope.WriteString ":BUS1:BIT1 ON"
```

## :BUS<n>:BITS

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:BUS&lt;n&gt;:BITS &lt;channel_list&gt;, &lt;display&gt;</code>
	<code>&lt;channel_list&gt; ::= (@&lt;m&gt;,&lt;m&gt;:&lt;m&gt;, ...)</code> where commas separate bits and colons define bit ranges.
	<code>&lt;m&gt; ::= An integer, 0,...,15, defines a digital channel affected by the command.</code>
	<code>&lt;display&gt; ::= {{1   ON}   {0   OFF}}</code>
	<code>&lt;n&gt; ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.</code>

The :BUS<n>:BITS command includes or excludes the selected bits in the channel list in the definition of the selected bus. If the parameter is a 1 (ON) then the bits in the channel list are included as part of the selected bus definition. If the parameter is a 0 (OFF) then the bits in the channel list are excluded from the definition of the selected bus.

### NOTE

This command is only valid for the MSO models.

<b>Query Syntax</b>	<code>:BUS&lt;n&gt;:BITS?</code>
	The :BUS<n>:BITS? query returns the definition for the specified bus.
<b>Return Format</b>	<code>&lt;channel_list&gt;, &lt;display&gt;&lt;NL&gt;</code>
	<code>&lt;channel_list&gt; ::= (@&lt;m&gt;,&lt;m&gt;:&lt;m&gt;, ...)</code> where commas separate bits and colons define bit ranges.
	<code>&lt;display&gt; ::= {0   1}</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :BUS&lt;n&gt; Commands</a>" on page 258</li> <li>• "<a href="#">":BUS&lt;n&gt;:BIT&lt;m&gt;"</a> on page 259</li> <li>• "<a href="#">":BUS&lt;n&gt;:CLEar"</a> on page 262</li> <li>• "<a href="#">":BUS&lt;n&gt;:DISPlay"</a> on page 263</li> <li>• "<a href="#">":BUS&lt;n&gt;:LABEL"</a> on page 264</li> <li>• "<a href="#">":BUS&lt;n&gt;:MASK"</a> on page 265</li> </ul>
<b>Example Code</b>	<pre>' Include digital channels 1, 2, 4, 5, 6, 7, 8, and 9 in bus 1: myScope.WriteString ":BUS1:BITS (@1,2,4:9), ON"  ' Include digital channels 1, 5, 7, and 9 in bus 1: myScope.WriteString ":BUS1:BITS (@1,5,7,9), ON"  ' Include digital channels 1 through 15 in bus 1: myScope.WriteString ":BUS1:BITS (@1:15), ON"</pre>

```
' Include digital channels 1 through 5, 8, and 14 in bus 1:  
myScope.WriteString ":BUS1:BITS (@1:5,8,14), ON"
```

**:BUS<n>:CLEar****N** (see [page 1334](#))**Command Syntax**    `:BUS<n>:CLEar`

`<n>` ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:CLEar command excludes all of the digital channels from the selected bus definition.

**NOTE**

This command is only valid for the MSO models.

**See Also**

- "[Introduction to :BUS<n> Commands](#)" on page 258
- "[":BUS<n>:BIT<m>"](#) on page 259
- "[":BUS<n>:BITS"](#) on page 260
- "[":BUS<n>:DISPlay"](#) on page 263
- "[":BUS<n>:LABEL"](#) on page 264
- "[":BUS<n>:MASK"](#) on page 265

## :BUS<n>:DISPlay

**N** (see [page 1334](#))

**Command Syntax** :BUS<n>:DISPlay <value>

<value> ::= {{1 | ON} | {0 | OFF}}

<n> ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:DISPlay command enables or disables the view of the selected bus.

### NOTE

This command is only valid for the MSO models.

**Query Syntax** :BUS<n>:DISPlay?

The :BUS<n>:DISPlay? query returns the display value of the selected bus.

**Return Format** <value><NL>

<value> ::= {0 | 1}

**See Also** • "[Introduction to :BUS<n> Commands](#)" on page 258

- "[:BUS<n>:BIT<m>](#)" on page 259
- "[:BUS<n>:BITS](#)" on page 260
- "[:BUS<n>:CLEar](#)" on page 262
- "[:BUS<n>:LABEL](#)" on page 264
- "[:BUS<n>:MASK](#)" on page 265

## :BUS<n>:LABEL

**N** (see [page 1334](#))

**Command Syntax**    `:BUS<n>:LABEL <quoted_string>`

`<quoted_string>` ::= any series of 10 or less characters as a quoted ASCII string.

`<n>` ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:LABEL command sets the bus label to the quoted string. Setting a label for a bus will also result in the name being added to the label list.

### NOTE

This command is only valid for the MSO models.

### NOTE

Label strings are 10 characters or less, and may contain any commonly used ASCII characters. Labels with more than 10 characters are truncated to 10 characters.

### Query Syntax

`:BUS<n>:LABEL?`

The :BUS<n>:LABEL? query returns the name of the specified bus.

### Return Format

`<quoted_string><NL>`

`<quoted_string>` ::= any series of 10 or less characters as a quoted ASCII string.

### See Also

- ["Introduction to :BUS<n> Commands"](#) on page 258
- [":BUS<n>:BIT<m>"](#) on page 259
- [":BUS<n>:BITS"](#) on page 260
- [":BUS<n>:CLEAR"](#) on page 262
- [":BUS<n>:DISPLAY"](#) on page 263
- [":BUS<n>:MASK"](#) on page 265
- [":CHANnel<n>:LABEL"](#) on page 287
- [":DISPLAY:LABELList"](#) on page 342
- [":DIGital<d>:LABEL"](#) on page 324

### Example Code

```
' Set the bus 1 label to "Data":  
myScope.WriteString ":BUS1:LABEL 'Data'"
```

## :BUS<n>:MASK

**N** (see [page 1334](#))

**Command Syntax** :BUS<n>:MASK <mask>

```
<mask> ::= 32-bit integer in decimal, <nondecimal>, or <string>
<nondecimal> ::= #Hnn...n where n ::= {0,...,9 | A,...,F} for hexadecimal
<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal
<n> ::= An integer, 1 or 2, is attached as a suffix to BUS
and defines the bus that is affected by the command.
```

The :BUS<n>:MASK command defines the bits included and excluded in the selected bus according to the mask. Set a mask bit to a "1" to include that bit in the selected bus, and set a mask bit to a "0" to exclude it.

### NOTE

This command is only valid for the MSO models.

**Query Syntax** :BUS<n>:MASK?

The :BUS<n>:MASK? query returns the mask value for the specified bus.

**Return Format** <mask><NL> in decimal format

- See Also**
- "[Introduction to :BUS<n> Commands](#)" on page 258
  - "[":BUS<n>:BIT<m>"](#) on page 259
  - "[":BUS<n>:BITS"](#) on page 260
  - "[":BUS<n>:CLEar"](#) on page 262
  - "[":BUS<n>:DISPlay"](#) on page 263
  - "[":BUS<n>:LABEL"](#) on page 264



# 9 :CALibrate Commands

Utility commands for viewing calibration status and for starting the user calibration procedure. See "[Introduction to :CALibrate Commands](#)" on page 268.

**Table 90** :CALibrate Commands Summary

Command	Query	Options and Query Returns
n/a	:CALibrate:DATE? (see <a href="#">page 269</a> )	<return value> ::= <year>,<month>,<day>; all in NR1 format
:CALibrate:LABEL <string> (see <a href="#">page 270</a> )	:CALibrate:LABEL? (see <a href="#">page 270</a> )	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:OUTPut <signal> (see <a href="#">page 271</a> )	:CALibrate:OUTPut? (see <a href="#">page 271</a> )	<signal> ::= {TRIGgers   MASK   WAVEgen   WGEN1   WGEN2} Note: WAVE and WGEN1 are equivalent. Note: WGEN2 only available on models with 2 WaveGen outputs.
n/a	:CALibrate:PROTected? (see <a href="#">page 273</a> )	{PROTected   UNPROtected}
:CALibrate:START (see <a href="#">page 274</a> )	n/a	n/a
n/a	:CALibrate:STATUS? (see <a href="#">page 275</a> )	<return value> ::= <status_code>,<status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:TEMPeratur e? (see <a href="#">page 276</a> )	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see <a href="#">page 277</a> )	<return value> ::= <hours>,<minutes>,<seconds>; all in NR1 format

- Introduction to :CALibrate Commands**
- The CALibrate subsystem provides utility commands for:
- Determining the state of the calibration factor protection switch (CAL PROTECT).
  - Saving and querying the calibration label string.
  - Reporting the calibration time and date.
  - Reporting changes in the temperature since the last calibration.
  - Starting the user calibration procedure.

## :CALibrate:DATE

**N** (see [page 1334](#))

**Query Syntax** :CALibrate:DATE?

The :CALibrate:DATE? query returns the date of the last calibration.

**Return Format** <date><NL>

<date> ::= year,month,day in NR1 format<NL>

**See Also** • ["Introduction to :CALibrate Commands"](#) on page 268

## :CALibrate:LABEL

**N** (see [page 1334](#))

**Command Syntax** :CALibrate:LABEL <string>

<string> ::= quoted ASCII string of up to 32 characters in length,  
not including the quotes

The CALibrate:LABEL command saves a string that is up to 32 characters in length into the instrument's non-volatile memory. The string may be used to record calibration dates or other information as needed.

**Query Syntax** :CALibrate:LABEL?

The :CALibrate:LABEL? query returns the contents of the calibration label string.

**Return Format** <string><NL>

<string> ::= unquoted ASCII string of up to 32 characters in length

**See Also** · "Introduction to :CALibrate Commands" on page 268

## :CALibrate:OUTPut

**N** (see [page 1334](#))

### Command Syntax

```
:CALibrate:OUTPut <signal>
<signal> ::= {TRIGgers | MASK | WAVEgen | WGEN1 | NFC}
```

Note: WAVE and WGEN1 are equivalent.

The CALibrate:OUTPut command sets the signal that is available on the rear panel TRIG OUT BNC:

- TRIGgers – pulse when a trigger event occurs.
- MASK – signal from mask test indicating a failure.
- WAVEgen, WGEN1 – waveform generator sync output signal. This signal depends on the :WGEN<w>:FUNCTION setting:

Waveform Type	Sync Signal Characteristics
SINusoid, SQUare, RAMP,PULSe, SINC, EXPRIse, EXPFall, GAUSSian	The Sync signal is a TTL positive pulse that occurs when the waveform rises above zero volts (or the DC offset value).
DC, NOISe, CARDiac	N/A

- NFC – This option is available in the Near Field Communication (NFC) trigger mode when the ATRigger (Arm & Trigger) trigger event is selected (see [":TRIGger:NFC:TEvent" on page 1091](#)). The ATRigger trigger event lets you arm the oscilloscope on one event and then trigger on a second event or after a specified timeout period if the second event does not occur.

When :CALibrate:OUTPut is NFC and the specified event arms, the *TRIG OUTBNC* goes high. The oscilloscope waits until a second event is found or until the specified timeout period expires and then triggers.

- For NFC-A, the second event is SDD\_REQ.
- For NFC-B, the second event is ATTRIB.
- For NFC-F, the second event is ATR\_REQ.

When the oscilloscope triggers, the *TRIG OUTBNC* line goes low.

### Query Syntax

```
:CALibrate:OUTPut ?
```

The :CALibrate:OUTPut query returns the current source of the TRIG OUT BNC signal.

**Return Format** <signal><NL>

<signal> ::= {TRIG | MASK | WAVE | NFC}

**See Also** • "[Introduction to :CALibrate Commands](#)" on page 268  
• "[":WGEN<w>:FUNCTION](#)" on page 1193

## :CALibrate:PROTected

**N** (see [page 1334](#))

**Query Syntax** :CALibrate:PROTected?

The :CALibrate:PROTected? query returns the rear-panel calibration protect (CAL PROTECT) button state. The value PROTected indicates calibration is disabled, and UNPROTected indicates calibration is enabled.

**Return Format** <switch><NL>

<switch> ::= {PROTected | UNPROTected}

**See Also** • ["Introduction to :CALibrate Commands"](#) on page 268

## :CALibrate:STARt

**N** (see [page 1334](#))

**Command Syntax** :CALibrate:STARt

The CALibrate:STARt command starts the user calibration procedure.

### NOTE

Before starting the user calibration procedure, you must set the rear panel CALIBRATION switch to UNPROTECTED, and you must connect BNC cables from the TRIG OUT connector to the analog channel inputs. See the *User's Guide* for details.

### See Also

- "[Introduction to :CALibrate Commands](#)" on page 268
- "[":CALibrate:PROTected](#)" on page 273

## :CALibrate:STATus

**N** (see [page 1334](#))

### Query Syntax

`:CALibrate:STATus?`

The :CALibrate:STATus? query returns the summary results of the last user calibration procedure.

### Return Format

```
<return value><NL>
<return value> ::= <status_code>,<status_string>
<status_code> ::= an integer status code
<status_string> ::= an ASCII status string
```

The status codes and strings can be:

Status Code	Status String
+0	Calibrated
-1	Not Calibrated

### See Also

- "Introduction to :CALibrate Commands" on page 268

## :CALibrate:TEMPerature

**N** (see [page 1334](#))

**Query Syntax** :CALibrate:TEMPerature?

The :CALibrate:TEMPerature? query returns the change in temperature since the last user calibration procedure.

**Return Format** <return value><NL>

<return value> ::= degrees C delta since last cal in NR3 format

**See Also** · "Introduction to :CALibrate Commands" on page 268

## :CALibrate:TIME

**N** (see [page 1334](#))

**Query Syntax** :CALibrate:TIME?

The :CALibrate:TIME? query returns the time of the last calibration.

**Return Format** <date><NL>

<date> ::= hour,minutes,seconds in NR1 format

**See Also** • ["Introduction to :CALibrate Commands"](#) on page 268



# 10 :CHANnel<n> Commands

Control all oscilloscope functions associated with individual analog channels or groups of channels. See "[Introduction to :CHANnel<n> Commands](#)" on page 281.

**Table 91** :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BWLimit { {0   OFF}   {1   ON}} (see <a href="#">page 282</a> )	:CHANnel<n>:BWLimit? (see <a href="#">page 282</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:COUPling <coupling> (see <a href="#">page 283</a> )	:CHANnel<n>:COUPling? (see <a href="#">page 283</a> )	<coupling> ::= {AC   DC} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:DISPlay { {0   OFF}   {1   ON}} (see <a href="#">page 284</a> )	:CHANnel<n>:DISPlay? (see <a href="#">page 284</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:IMPedance <impedance> (see <a href="#">page 285</a> )	:CHANnel<n>:IMPedance? (see <a href="#">page 285</a> )	<impedance> ::= {ONEMeg   FIFTy} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:INVert { {0   OFF}   {1   ON}} (see <a href="#">page 286</a> )	:CHANnel<n>:INVert? (see <a href="#">page 286</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:LABEL <string> (see <a href="#">page 287</a> )	:CHANnel<n>:LABEL? (see <a href="#">page 287</a> )	<string> ::= any series of 32 or less ASCII characters enclosed in quotation marks <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:OFFSet <offset>[suffix] (see <a href="#">page 288</a> )	:CHANnel<n>:OFFSet? (see <a href="#">page 288</a> )	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V   mV} <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe <attenuation> (see <a href="#">page 289</a> )	:CHANnel<n>:PROBe? (see <a href="#">page 289</a> )	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4r in NR1 format

**Table 91** :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe:HEA D[:TYPE] <head_param> (see <a href="#">page 290</a> )	:CHANnel<n>:PROBe:HEA D[:TYPE]? (see <a href="#">page 290</a> )	<head_param> ::= {SEND0   SEND6   SEND12   SEND20   DIFF0   DIFF6   DIFF12   DIFF20   NONE} <n> ::= 1 to (# analog channels) in NR1 format
n/a	:CHANnel<n>:PROBe:ID? (see <a href="#">page 291</a> )	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:MMO Del <value> (see <a href="#">page 292</a> )	:CHANnel<n>:PROBe:MMO Del? (see <a href="#">page 292</a> )	<value> ::= {P5205   P5210   P6205   P6241   P6243   P6245   P6246   P6247   P6248   P6249   P6250   P6251   P670X   P671X   TCP202} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:SKE W <skew_value> (see <a href="#">page 293</a> )	:CHANnel<n>:PROBe:SKE W? (see <a href="#">page 293</a> )	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:STY Pe <signal type> (see <a href="#">page 294</a> )	:CHANnel<n>:PROBe:STY Pe? (see <a href="#">page 294</a> )	<signal type> ::= {DIFFerential   SINGle} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROTectio n (see <a href="#">page 295</a> )	:CHANnel<n>:PROTectio n? (see <a href="#">page 295</a> )	{NORM   TRIP} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:RANGE <range>[suffix] (see <a href="#">page 296</a> )	:CHANnel<n>:RANGE? (see <a href="#">page 296</a> )	<range> ::= Vertical full-scale range value in NR3 format [suffix] ::= {V   mV} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:SCALe <scale>[suffix] (see <a href="#">page 297</a> )	:CHANnel<n>:SCALe? (see <a href="#">page 297</a> )	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V   mV} <n> ::= 1 to (# analog channels) in NR1 format

**Table 91** :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:UNITS <units> (see <a href="#">page 298</a> )	:CHANnel<n>:UNITS? (see <a href="#">page 298</a> )	<units> ::= {VOLT   AMPere} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:VERNier { {0   OFF}   {1   ON} } (see <a href="#">page 299</a> )	:CHANnel<n>:VERNier? (see <a href="#">page 299</a> )	{0   1} <n> ::= 1 to (# analog channels) in NR1 format

**Introduction to  
:CHANnel<n>  
Commands**

<n> ::= 1 to (# analog channels) in NR1 format

The CHANnel<n> subsystem commands control an analog channel (vertical or Y-axis of the oscilloscope). Channels are independently programmable for all offset, probe, coupling, bandwidth limit, inversion, vernier, and range (scale) functions. The channel number (1, 2, 3, or 4) specified in the command selects the analog channel that is affected by the command.

A label command provides identifying annotations of up to 10 characters.

You can toggle the channel displays on and off with the :CHANnel<n>:DISPlay command as well as with the root level commands :VIEW and :BLANK.

**NOTE**

The obsolete CHANnel subsystem is supported.

**Reporting the Setup**

Use :CHANnel1?, :CHANnel2?, :CHANnel3? or :CHANnel4? to query setup information for the CHANnel<n> subsystem.

**Return Format**

The following are sample responses from the :CHANnel<n>? query. In this case, the query was issued following a \*RST command.

```
:CHAN1:RANG +40.0E+00;OFFS +0.00000E+00;COUP DC;IMP ONEM;DISP 1;BWL 0;
INV 0;LAB "1";UNIT VOLT;PROB +10E+00;PROB:SKEW +0.00E+00;STYP SING
```

**:CHANnel<n>:BWLimit****C** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:BWLimit <bwlimit>`    `<bwlimit> ::= {{1 | ON} | {0 | OFF}}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:BWLimit command controls an internal low-pass filter. When the filter is on, the bandwidth of the specified channel is limited to approximately 25 MHz.

**Query Syntax**    `:CHANnel<n>:BWLimit?`

The :CHANnel<n>:BWLimit? query returns the current setting of the low-pass filter.

**Return Format**    `<bwlimit><NL>`    `<bwlimit> ::= {1 | 0}`**See Also**    • "Introduction to :CHANnel<n> Commands" on page 281

**:CHANnel<n>:COUpling****C** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:COUpling <coupling>`    `<coupling> ::= {AC | DC}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:COUpling command selects the input coupling for the specified channel. The coupling for each analog channel can be set to AC or DC.

**Query Syntax**    `:CHANnel<n>:COUpling?`

The :CHANnel<n>:COUpling? query returns the current coupling for the specified channel.

**Return Format**    `<coupling value><NL>`    `<coupling value> ::= {AC | DC}`**See Also**

- "Introduction to :CHANnel<n> Commands" on page 281

**:CHANnel<n>:DISPlay**(see [page 1334](#))**Command Syntax**    `:CHANnel<n>:DISPlay <display value>`    `<display value> ::= {{1 | ON} | {0 | OFF}}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:DISPlay command turns the display of the specified channel on or off.

**Query Syntax**    `:CHANnel<n>:DISPlay?`

The :CHANnel<n>:DISPlay? query returns the current display setting for the specified channel.

**Return Format**    `<display value><NL>`    `<display value> ::= {1 | 0}`**See Also**

- "Introduction to :CHANnel<n> Commands" on page 281

- "[:VIEW](#)" on page 242

- "[:BLANK](#)" on page 214

- "[:STATus](#)" on page 239

- "[:POD<n>:DISPlay](#)" on page 603

- "[:DIGital<d>:DISPlay](#)" on page 323

## :CHANnel<n>:IMPedance

**C** (see [page 1334](#))

**Command Syntax** :CHANnel<n>:IMPedance <impedance>

<impedance> ::= {ONEMeg | FIFTy}

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:IMPedance command selects the input impedance setting for the specified analog channel. The legal values for this command are ONEMeg (1 MΩ) and FIFTy (50Ω).

**Query Syntax** :CHANnel<n>:IMPedance?

The :CHANnel<n>:IMPedance? query returns the current input impedance setting for the specified channel.

**Return Format** <impedance value><NL>

<impedance value> ::= {ONEM | FIFT}

**See Also** • "Introduction to :CHANnel<n> Commands" on page 281

**:CHANnel<n>:INVert****N** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:INVert <invert value>`    `<invert value> ::= {{1 | ON} | {0 | OFF}}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:INVert command selects whether or not to invert the input signal for the specified channel. The inversion may be 1 (ON/inverted) or 0 (OFF/not inverted).

**Query Syntax**    `:CHANnel<n>:INVert?`

The :CHANnel<n>:INVert? query returns the current state of the channel inversion.

**Return Format**    `<invert value><NL>`    `<invert value> ::= {0 | 1}`**See Also**

- "Introduction to :CHANnel<n> Commands" on page 281

## :CHANnel<n>:LABel

**N** (see [page 1334](#))

**Command Syntax**

```
:CHANnel<n>:LABel <string>
<string> ::= quoted ASCII string
<n> ::= 1 to (# analog channels) in NR1 format
```

### NOTE

Label strings are 32 characters or less, and may contain any commonly used ASCII characters. Labels with more than 32 characters are truncated to 32 characters. Lower case characters are converted to upper case.

The :CHANnel<n>:LABel command sets the analog channel label to the string that follows. Setting a label for a channel also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

**Query Syntax**

```
:CHANnel<n>:LABel?
```

The :CHANnel<n>:LABel? query returns the label associated with a particular analog channel.

**Return Format**

```
<string><NL>
<string> ::= quoted ASCII string
```

**See Also**

- "[Introduction to :CHANnel<n> Commands](#)" on page 281
- "[":DISPlay:LABel](#)" on page 341
- "[":DIGItal<d>:LABel](#)" on page 324
- "[":DISPlay:LABList](#)" on page 342
- "[":BUS<n>:LABel](#)" on page 264

**Example Code**

```
' LABEL - This command allows you to write a name (32 characters
' maximum) next to the channel number. It is not necessary, but
' can be useful for organizing the display.
myScope.WriteString ":CHANnel1:LABel ""CAL 1""    ' Label ch1 "CAL 1".
myScope.WriteString ":CHANnel2:LABel ""CAL2""    ' Label ch1 "CAL2".
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :CHANnel<n>:OFFSet



(see [page 1334](#))

**Command Syntax**

```
:CHANnel<n>:OFFSet <offset> [<suffix>]
<offset> ::= Vertical offset value in NR3 format
<suffix> ::= {V | mV}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :CHANnel<n>:OFFSet command sets the value that is represented at center screen for the selected channel. The range of legal values varies with the value set by the :CHANnel<n>:RANGE and :CHANnel<n>:SCALE commands. If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value. Legal values are affected by the probe attenuation setting.

**Query Syntax**

```
:CHANnel<n>:OFFSet?
```

The :CHANnel<n>:OFFSet? query returns the current offset value for the selected channel.

**Return Format**

```
<offset><NL>
<offset> ::= Vertical offset value in NR3 format
```

**See Also**

- "[Introduction to :CHANnel<n> Commands](#)" on page 281
- "[":CHANnel<n>:RANGE](#)" on page 296
- "[":CHANnel<n>:SCALE](#)" on page 297
- "[":CHANnel<n>:PROBe](#)" on page 289

## :CHANnel<n>:PROBe



(see [page 1334](#))

<b>Command Syntax</b>	<pre>:CHANnel&lt;n&gt;:PROBe &lt;attenuation&gt; &lt;attenuation&gt; ::= probe attenuation ratio in NR3 format &lt;n&gt; ::= 1 to (# analog channels) in NR1 format The obsolete attenuation values X1, X10, X20, X100 are also supported.</pre>
	<p>The :CHANnel&lt;n&gt;:PROBe command specifies the probe attenuation factor for the selected channel.</p>
	<p>The probe attenuation factor may be from 0.1 to 10000.</p>
	<p>This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors, for making automatic measurements, and for setting trigger levels.</p>
	<p>If an AutoProbe probe is connected to the oscilloscope, the attenuation value cannot be changed from the sensed value. Attempting to set the oscilloscope to an attenuation value other than the sensed value produces an error.</p>
<b>Query Syntax</b>	<pre>:CHANnel&lt;n&gt;:PROBe?</pre>
	<p>The :CHANnel&lt;n&gt;:PROBe? query returns the current probe attenuation factor for the selected channel.</p>
<b>Return Format</b>	<pre>&lt;attenuation&gt;&lt;NL&gt; &lt;attenuation&gt; ::= probe attenuation ratio in NR3 format</pre>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• <a href="#">"Introduction to :CHANnel&lt;n&gt; Commands"</a> on page 281</li> <li>• <a href="#">":CHANnel&lt;n&gt;:RANGE"</a> on page 296</li> <li>• <a href="#">":CHANnel&lt;n&gt;:SCALE"</a> on page 297</li> <li>• <a href="#">":CHANnel&lt;n&gt;:OFFSet"</a> on page 288</li> </ul>
<b>Example Code</b>	<pre>' CHANNEL_PROBE - Sets the probe attenuation factor for the selected ' channel. The probe attenuation factor may be set from 0.1 to 1000. myScope.WriteString ":CHANnel1:PROBe 10" ' Set Probe to 10:1.</pre>
	<p>See complete example programs at: <a href="#">Chapter 42, "Programming Examples,"</a> starting on <a href="#">page 1343</a></p>

## :CHANnel<n>:PROBe:HEAD[:TYPE]

 (see [page 1334](#))

### Command Syntax

#### NOTE

This command is valid only for the 113xA Series probes.

```
:CHANnel<n>:PROBe:HEAD [:TYPE] <head_param>
<head_param> ::= {SEND0 | SEND6 | SEND12 | SEND20 | DIFF0 | DIFF6
                  | DIFF12 | DIFF20 | NONE}
<n> ::= {1 | 2 | 3 | 4}
```

The :CHANnel<n>:PROBe:HEAD[:TYPE] command sets an analog channel probe head type and dB value. You can choose from:

- SEND0 – Single-ended, 0dB.
- SEND6 – Single-ended, 6dB.
- SEND12 – Single-ended, 12dB.
- SEND20 – Single-ended, 20dB.
- DIFF0 – Differential, 0dB.
- DIFF6 – Differential, 6dB.
- DIFF12 – Differential, 12dB.
- DIFF20 – Differential, 20dB.

**Query Syntax** :CHANnel<n>:PROBe:HEAD [:TYPE] ?

The :CHANnel<n>:PROBe:HEAD[:TYPE]? query returns the current probe head type setting for the selected channel.

**Return Format** <head\_param><NL>

```
<head_param> ::= {SEND0 | SEND6 | SEND12 | SEND20 | DIFF0 | DIFF6
                  | DIFF12 | DIFF20 | NONE}
```

**See Also**

- "[Introduction to :CHANnel<n> Commands](#)" on page 281
- "[":CHANnel<n>:PROBe"](#) on page 289
- "[":CHANnel<n>:PROBe:ID"](#) on page 291
- "[":CHANnel<n>:PROBe:SKEW"](#) on page 293
- "[":CHANnel<n>:PROBe:STYPe"](#) on page 294

## :CHANnel<n>:PROBe:ID



(see [page 1334](#))

**Query Syntax** :CHANnel<n>:PROBe:ID?

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:ID? query returns the type of probe attached to the specified oscilloscope channel.

**Return Format** <probe id><NL>

<probe id> ::= unquoted ASCII string up to 11 characters

Some of the possible returned values are:

- 1131A
- 1132A
- 1134A
- 1147A
- 1153A
- 1154A
- 1156A
- 1157A
- 1158A
- 1159A
- AutoProbe
- E2621A
- E2622A
- E2695A
- E2697A
- HP1152A
- HP1153A
- NONE
- Probe
- Unknown
- Unsupported

**See Also** · ["Introduction to :CHANnel<n> Commands"](#) on page 281

## :CHANnel<n>:PROBe:MMODel

**N** (see [page 1334](#))

**Command Syntax**    `:CHANnel<n>:PROBe:MMODel <value>`

```
<value> ::= {P5205 | P5210 | P6205 | P6241 | P6243 | P6245 | P6246
             | P6247 | P6248 | P6249 | P6250 | P6251 | P670X | P671X | TCP202}
```

`<n>` ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:MMODel command sets the model number of the supported Tektronix probe.

**Query Syntax**    `:CHANnel<n>:PROBe:MMODel?`

The :CHANnel<n>:PROBe:MMODel? query returns the model number setting.

**Return Format**    `<value><NL>`

```
<value> ::= {P5205 | P5210 | P6205 | P6241 | P6243 | P6245 | P6246
             | P6247 | P6248 | P6249 | P6250 | P6251 | P670X | P671X | TCP202}
```

**See Also**

- [":CHANnel<n>:PROBe:ID" on page 291](#)

## :CHANnel<n>:PROBe:SKEW



(see [page 1334](#))

**Command Syntax** :CHANnel<n>:PROBe:SKEW <skew value>

<skew value> ::= skew time in NR3 format

<skew value> ::= -100 ns to +100 ns

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:SKEW command sets the channel-to-channel skew factor for the specified channel. Each analog channel can be adjusted + or -100 ns for a total of 200 ns difference between channels. You can use the oscilloscope's probe skew control to remove cable-delay errors between channels.

**Query Syntax** :CHANnel<n>:PROBe:SKEW?

The :CHANnel<n>:PROBe:SKEW? query returns the current probe skew setting for the selected channel.

**Return Format** <skew value><NL>

<skew value> ::= skew value in NR3 format

**See Also** • ["Introduction to :CHANnel<n> Commands" on page 281](#)

## :CHANnel&lt;n&gt;:PROBe:STYPe

(see [page 1334](#))

## Command Syntax

## NOTE

This command is valid only for the 113xA Series probes.

```
:CHANnel<n>:PROBe:STYPe <signal type>
<signal type> ::= {DIFFerential | SINGLE}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :CHANnel<n>:PROBe:STYPe command sets the channel probe signal type (STYPe) to differential or single-ended when using the 113xA Series probes and determines how offset is applied.

When single-ended is selected, the :CHANnel<n>:OFFSet command changes the offset value of the probe amplifier. When differential is selected, the :CHANnel<n>:OFFSet command changes the offset value of the channel amplifier.

## Query Syntax

```
:CHANnel<n>:PROBe:STYPe?
```

The :CHANnel<n>:PROBe:STYPe? query returns the current probe signal type setting for the selected channel.

## Return Format

```
<signal type><NL>
<signal type> ::= {DIFF | SING}
```

## See Also

- ["Introduction to :CHANnel<n> Commands" on page 281](#)
- [":CHANnel<n>:OFFSet" on page 288](#)

## :CHANnel<n>:PROTection

**N** (see [page 1334](#))

**Command Syntax** :CHANnel<n>:PROTection[:CLEAR]

```
<n> ::= 1 to (# analog channels) in NR1 format | 4}
```

When the analog channel input impedance is set to  $50\Omega$ , the input channels are protected against overvoltage. When an overvoltage condition is sensed, the input impedance for the channel is automatically changed to  $1 M\Omega$ .

The :CHANnel<n>:PROTection[:CLEAR] command is used to clear (reset) the overload protection. It allows the channel to be used again in  $50\Omega$  mode after the signal that caused the overload has been removed from the channel input.

Reset the analog channel input impedance to  $50\Omega$  (see "[:CHANnel<n>:IMPedance](#)" on page 285) after clearing the overvoltage protection.

**Query Syntax** :CHANnel<n>:PROTection?

The :CHANnel<n>:PROTection query returns the state of the input protection for CHANnel<n>. If a channel input has experienced an overload, TRIP (tripped) will be returned; otherwise NORM (normal) is returned.

**Return Format** {NORM | TRIP}<NL>

- See Also**
- "[Introduction to :CHANnel<n> Commands](#)" on page 281
  - "[:CHANnel<n>:COUPLing](#)" on page 283
  - "[:CHANnel<n>:IMPedance](#)" on page 285
  - "[:CHANnel<n>:PROBe](#)" on page 289

## :CHANnel<n>:RANGE

(see [page 1334](#))

**Command Syntax**    `:CHANnel<n>:RANGE <range>[<suffix>]`

`<range>` ::= vertical full-scale range value in NR3 format

`<suffix>` ::= {V | mV}

`<n>` ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:RANGE command defines the full-scale vertical axis of the selected channel. When using 1:1 probe attenuation, legal values for the range are from 8 mV to 40 V.

If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

**Query Syntax**    `:CHANnel<n>:RANGE?`

The :CHANnel<n>:RANGE? query returns the current full-scale range setting for the specified channel.

**Return Format**    `<range_argument><NL>`

`<range_argument>` ::= vertical full-scale range value in NR3 format

- See Also**
- "[Introduction to :CHANnel<n> Commands](#)" on page 281
  - "[":CHANnel<n>:SCALe](#)" on page 297
  - "[":CHANnel<n>:PROBe](#)" on page 289

**Example Code**

```
' CHANNEL_RANGE - Sets the full scale vertical range in volts.  The
' range value is 8 times the volts per division.
myScope.WriteString ":CHANnel1:RANGE 8"    ' Set the vertical range to
8 volts.
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :CHANnel<n>:SCALe

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:CHANnel&lt;n&gt;:SCALe &lt;scale&gt;[&lt;suffix&gt;]</code>
	<code>&lt;scale&gt;</code> ::= vertical units per division in NR3 format
	<code>&lt;suffix&gt;</code> ::= {V   mV}
	<code>&lt;n&gt;</code> ::= 1 to (# analog channels) in NR1 format
	The :CHANnel<n>:SCALe command sets the vertical scale, or units per division, of the selected channel.
	If the probe attenuation is changed, the scale value is multiplied by the probe's attenuation factor.
<b>Query Syntax</b>	<code>:CHANnel&lt;n&gt;:SCALe?</code>
	The :CHANnel<n>:SCALe? query returns the current scale setting for the specified channel.
<b>Return Format</b>	<code>&lt;scale value&gt;&lt;NL&gt;</code>
	<code>&lt;scale value&gt;</code> ::= vertical units per division in NR3 format
<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :CHANnel&lt;n&gt; Commands</a>" on page 281</li> <li>• "<a href="#">":CHANnel&lt;n&gt;:RANGE</a>" on page 296</li> <li>• "<a href="#">":CHANnel&lt;n&gt;:PROBe</a>" on page 289</li> </ul>

**:CHANnel<n>:UNITS****N** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:UNITS <units>`    `<units> ::= {VOLT | AMPere}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:UNITS command sets the measurement units for the connected probe. Select VOLT for a voltage probe and select AMPere for a current probe. Measurement results, channel sensitivity, and trigger level will reflect the measurement units you select.

**Query Syntax**    `:CHANnel<n>:UNITS?`

The :CHANnel<n>:UNITS? query returns the current units setting for the specified channel.

**Return Format**    `<units><NL>`    `<units> ::= {VOLT | AMP}`**See Also**

- "Introduction to :CHANnel<n> Commands" on page 281
- ":CHANnel<n>:RANGE" on page 296
- ":CHANnel<n>:PROBe" on page 289
- ":EXTernal:UNITS" on page 357

**:CHANnel<n>:VERNier****N** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:VERNier <vernier value>`    `<vernier value> ::= {{1 | ON} | {0 | OFF}}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:VERNier command specifies whether the channel's vernier (fine vertical adjustment) setting is ON (1) or OFF (0).

**Query Syntax**    `:CHANnel<n>:VERNier?`

The :CHANnel<n>:VERNier? query returns the current state of the channel's vernier setting.

**Return Format**    `<vernier value><NL>`    `<vernier value> ::= {0 | 1}`**See Also**

- "Introduction to :CHANnel<n> Commands" on page 281



# 11 :COUNTER Commands

When the optional DSOXDVMCTR digital voltmeter and counter analysis feature is licensed, these commands control the counter feature. See "[Introduction to :COUNTER Commands](#)" on page 302.

**Table 92** :COUNTER Commands Summary

Command	Query	Options and Query Returns
n/a	:COUNTER:CURREnt? (see <a href="#">page 303</a> )	<value> ::= current counter value in NR3 format
:COUNTER:ENABLE {{0   OFF}   {1   ON}} (see <a href="#">page 304</a> )	:COUNTER:ENABLE? (see <a href="#">page 304</a> )	{0   1}
:COUNTER:MODE <mode> (see <a href="#">page 305</a> )	:COUNTER:MODE (see <a href="#">page 305</a> )	<mode> ::= {FREQuency   PERiod   TOTalize}
:COUNTER:NDIGits <value> (see <a href="#">page 306</a> )	:COUNTER:NDIGits (see <a href="#">page 306</a> )	<value> ::= 3 to 8 in NR1 format
:COUNTER:SOURce <source> (see <a href="#">page 307</a> )	:COUNTER:SOURce? (see <a href="#">page 307</a> )	<source> ::= {CHANnel<n>   TQEEvent} <n> ::= 1 to (# analog channels) in NR1 format
:COUNTER:TOTalize:CLEar (see <a href="#">page 308</a> )	n/a	n/a
:COUNTER:TOTalize:GATE:ENABLE {{0   OFF}   {1   ON}} (see <a href="#">page 309</a> )	:COUNTER:TOTalize:GATE:ENABLE? (see <a href="#">page 309</a> )	{0   1}
:COUNTER:TOTalize:GATE:POLarity <polarity> (see <a href="#">page 310</a> )	:COUNTER:TOTalize:GATE:POLarity? (see <a href="#">page 310</a> )	<polarity> ::= {{NEGative   FALLing}   {POSitive   RISing}}

**Table 92** :COUNter Commands Summary (continued)

Command	Query	Options and Query Returns
:COUNTER:TOTalize:GAT E:SOURce <source> (see <a href="#">page 311</a> )	:COUNTER:TOTalize:GAT E:SOURce? (see <a href="#">page 311</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:COUNTER:TOTalize:SLO Pe <slope> (see <a href="#">page 312</a> )	:COUNTER:TOTalize:SLO Pe? (see <a href="#">page 312</a> )	<slope> ::= {{NEGative   FALLing}   {POSitive   RISing}}

**Introduction to :COUNter Commands** The :COUNter subsystem provides commands to control the counter feature.

Use :COUNter? to query setup information for the COUNter subsystem.

#### Return Format

The following is a sample response from the :COUNter? query. In this case, the query was issued following the \*RST command.

```
:COUN:ENAB 0;SOUR CHAN1;MODE FREQ;NDIG 5
```

**:COUNter:CURRent****N** (see [page 1334](#))**Query Syntax** :COUNter:CURRent?

The :COUNter:CURRent? query returns the current counter value.

**Return Format** <value><NL>

<value> ::= current counter value in NR3 format

- See Also**
- "[:COUNter:ENABLE](#)" on page 304
  - "[:COUNter:MODE](#)" on page 305
  - "[:COUNter:NDIGits](#)" on page 306
  - "[:COUNter:SOURce](#)" on page 307

## :COUNter:ENABLE

**N** (see [page 1334](#))

**Command Syntax** :COUNter:ENABLE {{0 | OFF} | {1 | ON}}

The :COUNter:ENABLE command enables or disables the counter feature.

**Query Syntax** :COUNter:ENABLE?

The :COUNter:ENABLE? query returns whether the counter is enabled or disabled.

**Return Format** <off\_on><NL>

{0 | 1}

**See Also**

- "[:COUNter:CURRent](#)" on page 303
- "[:COUNter:MODE](#)" on page 305
- "[:COUNter:NDIGits](#)" on page 306
- "[:COUNter:SOURce](#)" on page 307

## :COUNter:MODE

**N** (see [page 1334](#))

**Command Syntax** :COUNter:MODE <mode>

<mode> ::= {FREQuency | PERiod | TOTalize}

The :COUNter:MODE command sets the counter mode:

- FREQuency – the cycles per second (Hz) of the signal.
- PERiod – the time periods of the signal's cycles.
- TOTalize – the count of edge events on the signal.

**Query Syntax** :COUNter:MODE

The :COUNter:MODE? query returns the counter mode setting.

**Return Format** <mode><NL>

<mode> ::= {FREQ | PER | TOT}

- See Also**
- "[:COUNter:CURRent](#)" on page 303
  - "[:COUNter:ENABLE](#)" on page 304
  - "[:COUNter:NDIGITS](#)" on page 306
  - "[:COUNter:SOURce](#)" on page 307
  - "[:COUNter:TOTalize:CLEar](#)" on page 308
  - "[:COUNter:TOTalize:GATE:ENABLE](#)" on page 309
  - "[:COUNter:TOTalize:GATE:POLarity](#)" on page 310
  - "[:COUNter:TOTalize:GATE:SOURce](#)" on page 311
  - "[:COUNter:TOTalize:SLOPe](#)" on page 312

## :COUNter:NDIGits

**N** (see [page 1334](#))

**Command Syntax**    `:COUNter:NDIGits <value>`

`<value> ::= 3 to 8 in NR1 format`

The :COUNter:NDIGits command sets the number of digits of resolution used for the frequency or period counter.

Higher resolutions require longer gate times, which cause the measurement times to be longer as well.

**Query Syntax**    `:COUNter:NDIGits`

The :COUNter:NDIGits? query returns the currently set number of digits of resolution.

**Return Format**    `<value><NL>`

`<value> ::= 3 to 8 in NR1 format`

- See Also**
- "[:COUNter:CURREnt](#)" on page 303
  - "[:COUNter:ENABLE](#)" on page 304
  - "[:COUNter:MODE](#)" on page 305
  - "[:COUNter:SOURce](#)" on page 307
  - "[:COUNter:TOTalize:CLEar](#)" on page 308
  - "[:COUNter:TOTalize:GATE:ENABLE](#)" on page 309
  - "[:COUNter:TOTalize:GATE:POLarity](#)" on page 310
  - "[:COUNter:TOTalize:GATE:SOURce](#)" on page 311
  - "[:COUNter:TOTalize:SLOPe](#)" on page 312

## :COUNter:SOURce

**N** (see [page 1334](#))

### Command Syntax

```
:COUNter:SOURce <source>
<source> ::= {CHANnel<n> | TQEEvent}
```

<n> ::= 1 to (# analog channels) in NR1 format

The :COUNter:SOURce command selects the waveform source that the counter measures. You can select one of the analog input channels or the trigger qualified event signal.

With the TQEEvent (trigger qualified event) source (available when the trigger mode is not EDGE), you can see how often trigger events are detected. This can be more often than when triggers actually occur, due to the oscilloscope's acquisition time or update rate capabilities. The TRIG OUT signal shows when triggers actually occur. Remember that the oscilloscope's trigger circuitry does not re-arm until the holdoff time occurs (:TRIGger:HOLDoff) and that the minimum holdoff time is 40 ns; therefore, the maximum trigger qualified event frequency that can be counted is 25 MHz.

### Query Syntax

```
:COUNter:SOURce?
```

The :COUNter:SOURce? query returns the currently set counter source channel.

### Return Format

```
<source><NL>
```

### See Also

- "[:COUNter:CURRent](#)" on page 303
- "[:COUNter:ENABLE](#)" on page 304
- "[:COUNter:MODE](#)" on page 305
- "[:COUNter:NDIGits](#)" on page 306

## :COUNter:TOTalize:CLEar

**N** (see [page 1334](#))

**Command Syntax** :COUNter:TOTalize:CLEar

The :COUNter:TOTalize:CLEar command zeros the edge event counter.

**See Also**

- "[:COUNter:CURRent](#)" on page 303
- "[:COUNter:ENABLE](#)" on page 304
- "[:COUNter:MODE](#)" on page 305
- "[:COUNter:NDIGits](#)" on page 306
- "[:COUNter:SOURce](#)" on page 307
- "[:COUNter:TOTalize:GATE:ENABLE](#)" on page 309
- "[:COUNter:TOTalize:GATE:POLarity](#)" on page 310
- "[:COUNter:TOTalize:GATE:SOURce](#)" on page 311
- "[:COUNter:TOTalize:SLOPe](#)" on page 312

## :COUNter:TOTalize:GATE:ENABLE

**N** (see [page 1334](#))

**Command Syntax** :COUNter:TOTalize:GATE:ENABLE {{0 | OFF} | {1 | ON}}

The :COUNter:TOTalize:GATE:ENABLE command enables or disables totalizer gating.

When totalizer gating is enabled, the totalizer only counts edges when a second gating signal polarity is true. The second gating signal can be one of the remaining analog channel inputs.

**Query Syntax** :COUNter:TOTalize:GATE:ENABLE?

The :COUNter:TOTalize:GATE:ENABLE? query returns whether totalizer gating is enabled or disabled.

**Return Format** <off\_on><NL>

{0 | 1}

**See Also**

- "[:COUNter:CURRent](#)" on page 303
- "[:COUNter:ENABLE](#)" on page 304
- "[:COUNter:MODE](#)" on page 305
- "[:COUNter:NDIGits](#)" on page 306
- "[:COUNter:SOURce](#)" on page 307
- "[:COUNter:TOTalize:CLEar](#)" on page 308
- "[:COUNter:TOTalize:GATE:POLarity](#)" on page 310
- "[:COUNter:TOTalize:GATE:SOURce](#)" on page 311
- "[:COUNter:TOTalize:SLOPe](#)" on page 312

## :COUNter:TOTalize:GATE:POLarity

**N** (see [page 1334](#))

**Command Syntax**    `:COUNter:TOTalize:GATE:POLarity <polarity>`

`<polarity> ::= {{NEGative | FALLing} | {POSitive | RISing}}`

The :COUNter:TOTalize:GATE:POLarity command specifies the gating signal condition under which totalizer edges are counted.

The gating signal is specified with the :COUNter:TOTalize:GATE:SOURce command.

**Query Syntax**    `:COUNter:TOTalize:GATE:POLarity?`

The :COUNter:TOTalize:GATE:POLarity? query returns the currently specified gating signal condition.

**Return Format**    `<polarity><NL>`

`<polarity> ::= {NEG | POS}`

- See Also**
- [":COUNter:CURRent"](#) on page 303
  - [":COUNter:ENABLE"](#) on page 304
  - [":COUNter:MODE"](#) on page 305
  - [":COUNter:NDIGits"](#) on page 306
  - [":COUNter:SOURce"](#) on page 307
  - [":COUNter:TOTalize:CLEar"](#) on page 308
  - [":COUNter:TOTalize:GATE:ENABLE"](#) on page 309
  - [":COUNter:TOTalize:GATE:SOURce"](#) on page 311
  - [":COUNter:TOTalize:SLOPe"](#) on page 312

## :COUNter:TOTalize:GATE:SOURce

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:COUNter:TOTalize:GATE:SOURce &lt;source&gt;</code> <code>&lt;source&gt; ::= CHANnel&lt;n&gt;</code> <code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	The :COUNter:TOTalize:GATE:SOURce command selects the analog channel that has the totalizer gating signal.
<b>Query Syntax</b>	<code>:COUNter:TOTalize:GATE:SOURce?</code>
	The :COUNter:TOTalize:GATE:SOURce? query returns the current totalizer gating signal source.
<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">":COUNter:CURREnt"</a> on page 303</li> <li>· <a href="#">":COUNter:ENABLE"</a> on page 304</li> <li>· <a href="#">":COUNter:MODE"</a> on page 305</li> <li>· <a href="#">":COUNter:NDIGits"</a> on page 306</li> <li>· <a href="#">":COUNter:SOURce"</a> on page 307</li> <li>· <a href="#">":COUNter:TOTalize:CLEar"</a> on page 308</li> <li>· <a href="#">":COUNter:TOTalize:GATE:ENABLE"</a> on page 309</li> <li>· <a href="#">":COUNter:TOTalize:GATE:POLarity"</a> on page 310</li> <li>· <a href="#">":COUNter:TOTalize:SLOPe"</a> on page 312</li> </ul>

**:COUNter:TOTalize:SLOPe****N** (see [page 1334](#))**Command Syntax**    `:COUNter:TOTalize:SLOPe <slope>``<slope> ::= {{NEGative | FALLing} | {POSitive | RISing}}`

The :COUNter:TOTalize:SLOPe command specifies whether positive or negative edges are counted.

**Query Syntax**    `:COUNter:TOTalize:SLOPe?`

The :COUNter:TOTalize:SLOPe? query returns the currently set slope specification.

**Return Format**    `<slope><NL>``<slope> ::= {NEG | POS}`

- See Also**
- "[:COUNter:CURRent](#)" on page 303
  - "[:COUNter:ENABLE](#)" on page 304
  - "[:COUNter:MODE](#)" on page 305
  - "[:COUNter:NDIGits](#)" on page 306
  - "[:COUNter:SOURce](#)" on page 307
  - "[:COUNter:TOTalize:CLEar](#)" on page 308
  - "[:COUNter:TOTalize:GATE:ENABLE](#)" on page 309
  - "[:COUNter:TOTalize:GATE:POLarity](#)" on page 310
  - "[:COUNter:TOTalize:GATE:SOURce](#)" on page 311

# 12 :DEMO Commands

When the education kit is licensed (Option EDU), you can output demonstration signals on the oscilloscope's Demo 1 and Demo 2 terminals. See "["Introduction to :DEMO Commands"](#) on page 313.

**Table 93** :DEMO Commands Summary

Command	Query	Options and Query Returns
:DEMO:FUNCTION <signal> (see <a href="#">page 314</a> )	:DEMO:FUNCTION? (see <a href="#">page 317</a> )	<signal> ::= {SINusoid   NOISy   PHASe   RINGing   SINGle   AM   CLK   GLITch   BURSt   MSO   RUNT   TRANSition   RFBurst   SHOLD   LFSine   FMBurst   ETE   CAN   LIN   UART   I2C   SPI   I2S   CANLin   ARINC   FLEXray   MIL   MIL2   USB   NMONotonic}
:DEMO:FUNCTION:PHASE: PHASE <angle> (see <a href="#">page 318</a> )	:DEMO:FUNCTION:PHASE: PHASE? (see <a href="#">page 318</a> )	<angle> ::= angle in degrees from 0 to 360 in NR3 format
:DEMO:OUTPut {{0   OFF}   {1   ON}} (see <a href="#">page 319</a> )	:DEMO:OUTPut? (see <a href="#">page 319</a> )	{0   1}

**Introduction to :DEMO Commands** The :DEMO subsystem provides commands to output demonstration signals on the oscilloscope's Demo 1 and Demo 2 terminals.

## Reporting the Setup

Use :DEMO? to query setup information for the DEMO subsystem.

## Return Format

The following is a sample response from the :DEMO? query. In this case, the query was issued following the \*RST command.

```
:DEMO:FUNC SIN;OUTP 0
```

## :DEMO:FUNCTION

**N** (see [page 1334](#))

**Command Syntax** :DEMO:FUNCTION <signal>

```
<signal> ::= {SINusoid | NOISy | PHASe | RINGing | SINGle | AM | CLK
               | GLITch | BURSt | MSO | RUNT | TRANSition | RFBurst
               | SHOLD | LFSine | FMBurst | ETE | CAN | LIN | UART
               | I2C | SPI | I2S | CANLin | ARINC | FLEXray | MIL
               | MIL2 | NMONotonic | DCMotor | HARMonics | COUpling
               | CFD | SENT | KEYSight}
```

The :DEMO:FUNCTION command selects the type of demo signal:

Demo Signal Function	Demo 1 Terminal	Demo 2 Terminal
SINusoid	5 MHz sine wave @ ~ 6 Vpp, 0 V offset	Off
NOISy	1 kHz sine wave @ ~ 2.4 Vpp, 0.0 V offset, with ~ 0.5 Vpp of random noise added	Off
PHASe	1 kHz sine wave @ 2.4 Vpp, 0.0 V offset	1 kHz sine wave @ 2.4 Vpp, 0.0 V offset , phase shifted by the amount entered using the " <a href="#">":DEMO:FUNCTION:PHASE:PHASE</a> " on page 318 command
RINGing	500 kHz digital pulse @ ~ 3 Vpp, 1.5 V offset, and ~500 ns pulse width with ringing	Off
SINGle	~500 ns wide digital pulse with ringing @ ~ 3 Vpp, 1.5 V offset Press the front panel <b>Set Off</b> <b>Single-Shot</b> softkey to cause the selected single-shot signal to be output.	Off
AM	26 kHz sine wave, ~ 7 Vpp, 0 V offset	Amplitude modulated signal, ~ 3 Vpp, 0 V offset, with ~13 MHz carrier and sine envelope
CLK	3.6 MHz clock @ ~2 Vpp, 1 V offset, with infrequent glitch (1 glitch per 1,000,000 clocks)	Off
GLITch	Burst of 6 digital pulses (plus infrequent glitch) that occurs once every 80 µs @ ~3.6 Vpp, ~1.8 V offset	Off

Demo Signal Function	Demo 1 Terminal	Demo 2 Terminal
BURSt	Burst of digital pulses that occur every 50 µs @ ~ 3.6 Vpp, ~1.5 V offset	Off
MSO	3.1 kHz stair-step sine wave output of DAC @ ~1.5 Vpp, 0.75 V offset DAC input signals are internally routed to digital channels D0 through D7	~3.1 kHz sine wave filtered from DAC output @ ~ 600 mVpp, 300 mV offset
RUNT	Digital pulse train with positive and negative runt pulses @ ~ 3.5 Vpp, 1.75 V offset	Off
TRANSition	Digital pulse train with two different edge speeds @ ~ 3.5 Vpp, 1.75 V offset	Off
RFBurst	5-cycle burst of a 10 MHz amplitude modulated sine wave @ ~ 2.6 Vpp, 0 V offset occurring once every 4 ms	Off
SHOLD	6.25 MHz digital clock @ ~ 3.5 Vpp, 1.75 V offset	Data signal @ ~3.5 Vpp, 1.75 V offset
LFSine	30 Hz sine wave @ ~2.7 Vpp, 0 V offset, with very narrow glitch near each positive peak	Off
FMBurst	FM burst, modulated from ~100 kHz to ~1 MHz, ~5.0 Vpp, ~600 mV offset.	Off
ETE	100 kHz pulse, 400 ns wide @ ~3.3 Vpp, 1.65 V offset	600 ns analog burst (@ ~3.3 Vpp, 0.7 V offset) followed by 3.6 µs digital burst @ ~3.3 Vpp, 1.65 V offset) at a 100 kHz repetitive rate
CAN	CAN_L, 125 kbps dominant-low, ~2.8 Vpp, ~1.4 V offset	Off
LIN	LIN, 19.2 kbs, ~2.8 Vpp, ~1.4 V offset	Off
UART	Receive data (RX) with odd parity, 19.2 kbps, 8-bit words, LSB out 1st, low idle @ ~2.8 Vpp, 1.4 V offset	Transmit data (TX) with odd parity, 19.2 kbps, 8-bit words, LSB out 1st, low idle @ ~ 2.8 Vpp, 1.4 V offset
I2C	I2C serial clock signal (SCL) @ ~2.8 Vpp, 1.4 V offset	I2C serial data signal (SDA) @ ~ 2.8 Vpp, 1.4 V offset

Demo Signal Function	Demo 1 Terminal	Demo 2 Terminal
SPI	<p>Off</p> <p>Signals are internally routed to digital channels D6 through D9:</p> <ul style="list-style-type: none"> <li>▪ D9 – MOSI, TTL level, with MSB out 1st (internally routed to digital input).</li> <li>▪ D8 – MISO, TTL level, with MSB out 1st (internally routed to digital input).</li> <li>▪ D7 – CLK, TTL level (internally routed to digital input).</li> <li>▪ D6 – ~CS, low-enable, TTL level (internally routed to digital input).</li> </ul>	Off
I2S	<p>Off</p> <p>Signals are internally routed to digital channels D7 through D9:</p> <ul style="list-style-type: none"> <li>▪ D9 – SDATA, TTL level, with "standard" alignment (internally routed to digital input).</li> <li>▪ D8 – SCLK, TTL level, (internally routed to digital input).</li> <li>▪ D7 – WS, TTL level, low for left channel, high for right channel (internally routed to digital input)</li> </ul>	Off
CANLin	CAN_L, 250 kbps dominant-low, ~2.8 Vpp, ~1.4 V offset	LIN, 19.2 kbps, ~2.8 Vpp, ~1.4 V offset
ARINC	ARINC 429, 100 kbps, ~5 Vpp, ~0 V offset.	Off
FLEXray	FlexRay @ 10 Mbps, ~2.8 Vpp, ~0 V offset	Off
MIL	MIL-STD-1553 RT to RT transfer, received ~1.3 Vpp, transmitted ~4.8 Vpp, 0 V offset	Off
MIL2	MIL-STD-1553 RT to RT transfer, received ~1.3 Vpp, transmitted ~4.8 Vpp, 0 V offset	MIL-STD-1553 RT to BC transfer, received ~1.3 Vpp, transmitted ~4.8 Vpp, 0 V offset
NMONotonic	Digital pulse train with infrequent non-monotonic rising edges @ ~ 2.85 Vpp, 1.42 V offset	Off
DCMotor	Output of DAC controlling a DC motor: 800 mV pulse, 1 $\mu$ s wide, every 10 $\mu$ s, runt pulse every 100 ms.	Off

Demo Signal Function	Demo 1 Terminal	Demo 2 Terminal
HARMonics	1 kHz sine wave @ ~3.5 Vpp, 0.0 V offset, with a ~2 kHz sine wave coupled in	Off
COUPLing	1 kHz square wave @ ~1 Vpp, 0.0 V offset, with a ~90 kHz sine wave with ~180 mVpp riding on top	Off
CFD	CAN FD, ~2.4 Vpp, ~-1.2 V offset, 500 kb/s standard baud rate (sample point at 80%), 10 Mb/s FD baud rate (sample point at 50%)	Off
SENT	SENT, ~2.7 Vpp, ~1.35 V offset, 3 µs clock period, 6 nibbles in a Fast Channel Message, Slow Channel Messages in enhanced format, idle state high, with pause pulse	Off
KEYSight	Series of positive and negative pulses, 125 µs wide, amplitude modulated, ~2.6 Vpp, ~0 V offset	Off

**Query Syntax** :DEMO:FUNCTION?

The :DEMO:FUNCTION? query returns the currently selected demo signal type.

**Return Format** <signal><NL>

```
<signal> ::= {SIN | NOIS | PHAS | RING | SING | AM | CLK | GLIT
              | BURS | MSO | RUNT | TRAN | RFB | SHOL | LFS | FMB
              | ETE | CAN | LIN | UART | I2C | SPI | I2S | CANL
              | ARIN | FLEX | MIL | MIL2 | NMON | DCM | HARM
              | COUP | CFD | SENT | KEYS}
```

**See Also** • "Introduction to :DEMO Commands" on page 313

**:DEMO:FUNCTION:PHASE:PHASE****N** (see [page 1334](#))**Command Syntax**    `:DEMO:FUNCTION:PHASE:PHASE <angle>``<angle> ::= angle in degrees from 0 to 360 in NR3 format`

For the phase shifted sine demo signals, the :DEMO:FUNCTION:PHASE:PHASE command specifies the phase shift in the second sine waveform.

**Query Syntax**    `:DEMO:FUNCTION:PHASE:PHASE?`

The :DEMO:FUNCTION:PHASE:PHASE? query returns the currently set phase shift.

**Return Format**    `<angle><NL>``<angle> ::= angle in degrees from 0 to 360 in NR3 format`**See Also**

- "[Introduction to :DEMO Commands](#)" on page 313
- "[":DEMO:FUNCTION"](#) on page 314

## :DEMO:OUTPut

**N** (see [page 1334](#))

**Command Syntax**    `:DEMO:OUTPut <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :DEMO:OUTPut command specifies whether the demo signal output is ON (1) or OFF (0).

**Query Syntax**    `:DEMO:OUTPut?`

The :DEMO:OUTPut? query returns the current state of the demo signal output setting.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- "Introduction to :DEMO Commands" on page 313
- ":DEMO:FUNCTION" on page 314



# 13 :DIGItal<d> Commands

Control all oscilloscope functions associated with individual digital channels. See "[Introduction to :DIGItal<d> Commands](#)" on page 322.

**Table 94** :DIGItal<d> Commands Summary

Command	Query	Options and Query Returns
:DIGItal<d>:DISPlay { {0   OFF}   {1   ON}} (see <a href="#">page 323</a> )	:DIGItal<d>:DISPlay? (see <a href="#">page 323</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format {0   1}
:DIGItal<d>:LABel <string> (see <a href="#">page 324</a> )	:DIGItal<d>:LABel? (see <a href="#">page 324</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <string> ::= any series of 10 or less ASCII characters enclosed in quotation marks
:DIGItal<d>:POSIon <position> (see <a href="#">page 325</a> )	:DIGItal<d>:POSITION? (see <a href="#">page 325</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <position> ::= 0-7 if display size = large, 0-15 if size = medium, 0-31 if size = small Returns -1 when there is no space to display the digital waveform.
:DIGItal<d>:SIZE <value> (see <a href="#">page 326</a> )	:DIGItal<d>:SIZE? (see <a href="#">page 326</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {SMAll   MEDium   LARGe}
:DIGItal<d>:THReShold <value>[suffix] (see <a href="#">page 327</a> )	:DIGItal<d>:THReShold? (see <a href="#">page 327</a> )	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {CMOS   ECL   TTL   <user defined value>} <user defined value> ::= value in NR3 format from -8.00 to +8.00 [suffix] ::= {V   mV   uV}

<b>Introduction to :DIGITAL&lt;d&gt; Commands</b>	<p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>The DIGITAL subsystem commands control the viewing, labeling, and positioning of digital channels. They also control threshold settings for groups of digital channels, or <i>pods</i>.</p>
---	---

**NOTE**

These commands are only valid for the MSO models.

---

### Reporting the Setup

Use :DIGITAL<d>? to query setup information for the DIGITAL subsystem.

### Return Format

The following is a sample response from the :DIGITAL0? query. In this case, the query was issued following a \*RST command.

```
:DIG0:DISP 0;THR +1.40E+00;LAB 'D0';POS +0
```

## :DIGItal<d>:DISPlay

**N** (see [page 1334](#))

**Command Syntax** :DIGItal<d>:DISPlay <display>

```
<d> ::= 0 to (# digital channels - 1) in NR1 format
<display> ::= {{1 | ON} | {0 | OFF}}
```

The :DIGItal<d>:DISPlay command turns digital display on or off for the specified channel.

### NOTE

This command is only valid for the MSO models.

**Query Syntax** :DIGItal<d>:DISPlay?

The :DIGItal<d>:DISPlay? query returns the current digital display setting for the specified channel.

**Return Format** <display><NL>

```
<display> ::= {0 | 1}
```

**See Also**

- "[Introduction to :DIGItal<d> Commands](#)" on page 322
- "[":POD<n>:DISPlay](#)" on page 603
- "[":CHANnel<n>:DISPlay](#)" on page 284
- "[":VIEW](#)" on page 242
- "[":BLANK](#)" on page 214
- "[":STATus](#)" on page 239

## :DIGital<d>:LABel

**N** (see [page 1334](#))

**Command Syntax**

```
:DIGital<d>:LABel <string>
<d> ::= 0 to (# digital channels - 1) in NR1 format
<string> ::= any series of 10 or less characters as quoted ASCII string.
```

The :DIGital<d>:LABel command sets the channel label to the string that follows. Setting a label for a channel also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

**NOTE**

This command is only valid for the MSO models.

**NOTE**

Label strings are 10 characters or less, and may contain any commonly used ASCII characters. Labels with more than 10 characters are truncated to 10 characters.

**Query Syntax**

```
:DIGital<d>:LABel?
```

The :DIGital<d>:LABel? query returns the name of the specified channel.

**Return Format**

```
<label string><NL>
<label string> ::= any series of 10 or less characters as a quoted
ASCII string.
```

**See Also**

- "[Introduction to :DIGital<d> Commands](#)" on page 322
- "[:CHANnel<n>:LABel](#)" on page 287
- "[:DISPlay:LABList](#)" on page 342
- "[:BUS<n>:LABel](#)" on page 264

## :DIGItal<d>:POsition

**N** (see [page 1334](#))

**Command Syntax** :DIGItal<d>:POsition <position>

```
<d> ::= 0 to (# digital channels - 1) in NR1 format
<position> ::= integer in NR1 format.
```

Channel Size	Position	Top	Bottom
Large	0-7	7	0
Medium	0-15	15	0
Small	0-31	31	0

The :DIGItal<d>:POsition command sets the position of the specified channel. Note that bottom positions might not be valid depending on whether digital buses, serial decode waveforms, or the zoomed time base are displayed.

### NOTE

This command is only valid for the MSO models.

### Query Syntax

:DIGItal<d>:POsition?

The :DIGItal<d>:POsition? query returns the position of the specified channel.

If the returned value is "-1", this indicates there is no space to display the digital waveform (for example, when all serial lanes, digital buses, and the zoomed time base are displayed).

### Return Format

<position><NL>

```
<position> ::= integer in NR1 format.
```

### See Also

- ["Introduction to :DIGItal<d> Commands" on page 322](#)

## :DIGital<d>:SIZE

**N** (see [page 1334](#))

**Command Syntax**    `:DIGital<d>:SIZE <value>`

```
<d> ::= 0 to (# digital channels - 1) in NR1 format
<value> ::= {SMALL | MEDium | LARGe}
```

The :DIGital<d>:SIZE command specifies the size of digital channels on the display. Sizes are set for all digital channels. Therefore, if you set the size on digital channel 0 (for example), the same size is set on all other as well.

### NOTE

This command is only valid for the MSO models.

**Query Syntax**    `:DIGital<d>:SIZE?`

The :DIGital<d>:SIZE? query returns the size setting for the specified digital channels.

**Return Format**    `<size_value><NL>`

```
<size_value> ::= {SMAL | MED | LARG}
```

**See Also**

- "[Introduction to :DIGital<d> Commands](#)" on page 322
- "[":POD<n>:SIZE](#)" on page 604
- "[":DIGital<d>:POStion](#)" on page 325

## :DIGItal<d>:THreshold

**N** (see [page 1334](#))

### Command Syntax :DIGItal<d>:THreshold <value>

```
<d> ::= 0 to (# digital channels - 1) in NR1 format
<value> ::= {CMOS | ECL | TTL | <user defined value>[<suffix>] }
<user defined value> ::= -8.00 to +8.00 in NR3 format
<suffix> ::= {v | mV | uV}
• TTL = 1.4V
• CMOS = 2.5V
• ECL = -1.3V
```

The :DIGItal<d>:THreshold command sets the logic threshold value for all channels in the same *pod* as the specified channel. The threshold is used for triggering purposes and for displaying the digital data as high (above the threshold) or low (below the threshold).

### NOTE

This command is only valid for the MSO models.

### Query Syntax :DIGItal<d>:THreshold?

The :DIGItal<d>:THreshold? query returns the threshold value for the specified channel.

### Return Format <value><NL>

```
<value> ::= threshold value in NR3 format
```

### See Also

- "[Introduction to :DIGItal<d> Commands](#)" on page 322
- "[":POD<n>:THreshold](#)" on page 605
- "[":TRIGger\[:EDGE\]:LEVel](#)" on page 1073



# 14 :DISPlay Commands

Control how waveforms, graticule, and text are displayed and written on the screen. See "[Introduction to :DISPlay Commands](#)" on page 330.

**Table 95** :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n> {{0   OFF}   {1   ON}} (see <a href="#">page 332</a> )	:DISPlay:ANNotation<n>? (see <a href="#">page 332</a> )	{0   1} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:BACKground <mode> (see <a href="#">page 333</a> )	:DISPlay:ANNotation<n>:BACKground? (see <a href="#">page 333</a> )	<mode> ::= {OPAQue   INVerted   TRANsparent} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:COLOR <color> (see <a href="#">page 334</a> )	:DISPlay:ANNotation<n>:COLOR? (see <a href="#">page 334</a> )	<color> ::= {CH1   CH2   CH3   CH4   DIG   MATH   REF   MARKer   WHITE   RED} <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:TEXT <string> (see <a href="#">page 335</a> )	:DISPlay:ANNotation<n>:TEXT? (see <a href="#">page 335</a> )	<string> ::= quoted ASCII string (up to 254 characters) <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:X1Position <value> (see <a href="#">page 336</a> )	:DISPlay:ANNotation<n>:X1Position? (see <a href="#">page 336</a> )	<value> ::= an integer from 0 to (800 - width of annotation) in NR1 format. <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:ANNotation<n>:Y1Position <value> (see <a href="#">page 337</a> )	:DISPlay:ANNotation<n>:Y1Position? (see <a href="#">page 337</a> )	<value> ::= an integer from 0 to (480 - height of annotation) in NR1 format. <n> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:CLEar (see <a href="#">page 338</a> )	n/a	n/a

**Table 95** :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:DISPlay:DATA? [<format>] [,] [<palett e>] (see <a href="#">page 339</a> )	<format> ::= {BMP   BMP8bit   PNG} <palette> ::= {COLOR   GRAYscale} <display data> ::= data in IEEE 488.2 # format
:DISPlay:INTensity:WA Veform <value> (see <a href="#">page 340</a> )	:DISPlay:INTensity:WA Veform? (see <a href="#">page 340</a> )	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:LABel {{0   OFF}   {1   ON}} (see <a href="#">page 341</a> )	:DISPlay:LABel? (see <a href="#">page 341</a> )	{0   1}
:DISPlay:LABList <binary block> (see <a href="#">page 342</a> )	:DISPlay:LABList? (see <a href="#">page 342</a> )	<binary block> ::= an ordered list of up to 75 labels, each 10 characters maximum, separated by newline characters
:DISPlay:MENU <menu> (see <a href="#">page 343</a> )	n/a	<menu> ::= {MASK   MEASure   SEGmented   LISTer   POWER}
:DISPlay:SIDebar <sidebar> (see <a href="#">page 344</a> )	n/a	<sidebar> ::= {SUMMARY   CURSors   MEASurements   DVM   NAVigate   CONTrols   EVENTS   COUNTER}
:DISPlay:PERStance <value> (see <a href="#">page 345</a> )	:DISPlay:PERStance? (see <a href="#">page 345</a> )	<value> ::= {MINimum   INFinite   <time>} <time> ::= seconds in in NR3 format from 100E-3 to 60E0
:DISPlay:VECTors {1   ON} (see <a href="#">page 346</a> )	:DISPlay:VECTors? (see <a href="#">page 346</a> )	1

**Introduction to :DISPlay Commands** The DISPlay subsystem is used to control the display storage and retrieval of waveform data, labels, and text. This subsystem allows the following actions:

- Clear the waveform area on the display.
- Turn vectors on or off.
- Set waveform persistence.
- Specify labels.
- Save and Recall display data.

### Reporting the Setup

Use :DISPlay? to query the setup information for the DISPlay subsystem.

### Return Format

The following is a sample response from the :DISPlay? query. In this case, the query was issued following a \*RST command.

```
:DISP:LAB 0;VECT 1;PERS MIN
```

**:DISPlay:ANNotation<n>****N** (see [page 1334](#))**Command Syntax**    `:DISPlay:ANNotation<n> <setting>`    `<setting> ::= {{1 | ON} | {0 | OFF}}`    `<n> ::= an integer from 1 to 10 in NR1 format.`

The `:DISPlay:ANNotation<n>` command turns the annotation on and off. When on, the annotation appears in the upper left corner of the oscilloscope's display.

The annotation is useful for documentation purposes, to add notes before capturing screens.

**Query Syntax**    `:DISPlay:ANNotation<n>?`

The `:DISPlay:ANNotation<n>?` query returns the annotation setting.

**Return Format**    `<value><NL>`    `<value> ::= {0 | 1}`

- See Also**
- [":DISPlay:ANNotation<n>:TEXT" on page 335](#)
  - [":DISPlay:ANNotation<n>:COLOR" on page 334](#)
  - [":DISPlay:ANNotation<n>:BACKGROUND" on page 333](#)
  - [":DISPlay:ANNotation<n>:X1Position" on page 336](#)
  - [":DISPlay:ANNotation<n>:Y1Position" on page 337](#)
  - ["Introduction to :DISPlay Commands" on page 330](#)

## :DISPlay:ANNotation<n>:BACKground

**N** (see [page 1334](#))

**Command Syntax**    `:DISPlay:ANNotation<n>:BACKground <mode>`

`<mode> ::= {OPAQue | INVerted | TRANsparent}`

`<n> ::= an integer from 1 to 10 in NR1 format.`

The :DISPlay:ANNotation<n>:BACKground command specifies the background of the annotation:

- OPAQue – the annotation has a solid background.
- INVerted – the annotation's foreground and background colors are switched.
- TRANsparent – the annotation has a transparent background.

**Query Syntax**    `:DISPlay:ANNotation<n>:BACKground?`

The :DISPlay:ANNotation<n>:BACKground? query returns the specified annotation background mode.

**Return Format**    `<mode><NL>`

`<mode> ::= {OPAQ | INV | TRAN}`

**See Also**

- "[:DISPlay:ANNotation<n>](#)" on page 332
- "[:DISPlay:ANNotation<n>:TEXT](#)" on page 335
- "[:DISPlay:ANNotation<n>:COLor](#)" on page 334
- "[:DISPlay:ANNotation<n>:X1Position](#)" on page 336
- "[:DISPlay:ANNotation<n>:Y1Position](#)" on page 337
- "[Introduction to :DISPlay Commands](#)" on page 330

## :DISPlay:ANNotation<n>:COLor

**N** (see [page 1334](#))

**Command Syntax**    `:DISPlay:ANNotation<n>:COLor <color>`

`<color> ::= {CH1 | CH2 | CH3 | CH4 | DIG | MATH | REF | MARKer | WHITe  
| RED}`

`<n> ::= an integer from 1 to 10 in NR1 format.`

The :DISPlay:ANNotation<n>:COLor command specifies the annotation color. You can choose white, red, or colors that match analog channels, digital channels, math waveforms, reference waveforms, or markers.

**Query Syntax**    `:DISPlay:ANNotation<n>:COLor?`

The :DISPlay:ANNotation<n>:COLor? query returns the specified annotation color.

**Return Format**    `<color><NL>`

`<color> ::= {CH1 | CH2 | CH3 | CH4 | DIG | MATH | REF | MARK | WHIT  
| RED}`

**See Also**

- "[":DISPlay:ANNotation<n>"](#) on page 332
- "[":DISPlay:ANNotation<n>:TEXT"](#) on page 335
- "[":DISPlay:ANNotation<n>:BACKground"](#) on page 333
- "[":DISPlay:ANNotation<n>:X1Position"](#) on page 336
- "[":DISPlay:ANNotation<n>:Y1Position"](#) on page 337
- "["Introduction to :DISPlay Commands"](#) on page 330

## :DISPlay:ANNotation<n>:TEXT

**N** (see [page 1334](#))

### Command Syntax

```
:DISPlay:ANNotation<n>:TEXT <string>
<string> ::= quoted ASCII string (up to 254 characters)
<n> ::= an integer from 1 to 10 in NR1 format.
```

The :DISPlay:ANNotation<n>:TEXT command specifies the annotation string. The annotation string can contain as many characters as will fit in the Edit Annotation box on the oscilloscope's screen, up to 254 characters.

You can include a carriage return in the annotation string using the characters "\n". Note that this is not a new line character but the actual "\" (backslash) and "n" characters in the string. Carriage returns lessen the number of characters available for the annotation string.

Use :DISPlay:ANNotation<n>:TEXT "" to remotely clear the annotation text. (Two sets of quote marks without a space between them creates a NULL string.)

### Query Syntax

```
:DISPlay:ANNotation<n>:TEXT?
```

The :DISPlay:ANNotation<n>:TEXT? query returns the specified annotation text.

When carriage returns are present in the annotation text, they are returned as the actual carriage return character (ASCII 0x0D).

### Return Format

```
<string><NL>
<string> ::= quoted ASCII string
```

### See Also

- "[:DISPlay:ANNotation<n>](#)" on page 332
- "[:DISPlay:ANNotation<n>:COLor](#)" on page 334
- "[:DISPlay:ANNotation<n>:BACKground](#)" on page 333
- "[:DISPlay:ANNotation<n>:X1Position](#)" on page 336
- "[:DISPlay:ANNotation<n>:Y1Position](#)" on page 337
- "[Introduction to :DISPlay Commands](#)" on page 330

**:DISPlay:ANNotation<n>:X1Position****N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:DISPlay:ANNotation&lt;n&gt;:X1Position &lt;value&gt;</code>  <code>&lt;value&gt; ::= an integer from 0 to (800 - width of annotation) in NR1 format.</code>  <code>&lt;n&gt; ::= an integer from 1 to 10 in NR1 format.</code>
<b>Query Syntax</b>	<code>:DISPlay:ANNotation&lt;n&gt;:X1Position?</code>  The <code>:DISPlay:ANNotation&lt;n&gt;:X1Position?</code> query returns the annotation's horizontal X1 position.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code>  <code>&lt;value&gt; ::= an integer from 0 to (800 - width of annotation) in NR1 format.</code>
<b>See Also</b>	<ul style="list-style-type: none"><li><a href="#">":DISPlay:ANNotation&lt;n&gt;:Y1Position"</a> on page 337</li><li><a href="#">":DISPlay:ANNotation&lt;n&gt;"</a> on page 332</li><li><a href="#">":DISPlay:ANNotation&lt;n&gt;:COLOr"</a> on page 334</li><li><a href="#">":DISPlay:ANNotation&lt;n&gt;:BACKground"</a> on page 333</li><li><a href="#">":DISPlay:ANNotation&lt;n&gt;:TEXT"</a> on page 335</li></ul>

## :DISPlay:ANNotation<n>:Y1Position

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:DISPlay:ANNotation&lt;n&gt;:Y1Position &lt;value&gt;</code>
	<code>&lt;value&gt;</code> ::= an integer from 0 to (480 - height of annotation) in NR1 format.
	<code>&lt;n&gt;</code> ::= an integer from 1 to 10 in NR1 format.
	The :DISPlay:ANNotation<n>:Y1Position command sets the annotation's vertical Y1 position.
<b>Query Syntax</b>	<code>:DISPlay:ANNotation&lt;n&gt;:Y1Position?</code>
	The :DISPlay:ANNotation<n>:Y1Position? query returns the annotation's vertical Y1 position.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code>
	<code>&lt;value&gt;</code> ::= an integer from 0 to (480 - height of annotation) in NR1 format.
<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">:DISPlay:ANNotation&lt;n&gt;:X1Position</a>" on page 336</li> <li>• "<a href="#">:DISPlay:ANNotation&lt;n&gt;</a>" on page 332</li> <li>• "<a href="#">:DISPlay:ANNotation&lt;n&gt;:COLor</a>" on page 334</li> <li>• "<a href="#">:DISPlay:ANNotation&lt;n&gt;:BACKground</a>" on page 333</li> <li>• "<a href="#">:DISPlay:ANNotation&lt;n&gt;:TEXT</a>" on page 335</li> </ul>

## :DISPlay:CLEar

**N** (see [page 1334](#))

**Command Syntax** :DISPlay:CLEar

The :DISPlay:CLEar command clears the display and resets all associated measurements. If the oscilloscope is stopped, all currently displayed data is erased. If the oscilloscope is running, all of the data for active channels and functions is erased; however, new data is displayed on the next acquisition.

**See Also** • ["Introduction to :DISPlay Commands"](#) on page 330

## :DISPlay:DATA

**N** (see [page 1334](#))

**Query Syntax**

```
:DISPlay:DATA? [<format>][,<palette>]
<format> ::= {BMP | BMP8bit | PNG}
<palette> ::= {COLOR | GRAYscale}
```

The :DISPlay:DATA? query reads screen image data. You can choose 24-bit BMP, 8-bit BMP8bit, or 24-bit PNG formats in color or grayscale.

If no format or palette option is specified, the screen image is returned in BMP, COLOR format.

Screen image data is returned in the IEEE-488.2 # binary block data format.

**Return Format**

```
<display data><NL>
<display data> ::= binary block data in IEEE-488.2 # format.
```

**See Also**

- "[Introduction to :DISPlay Commands](#)" on page 330
- "[":HARDcopy:INKSaver](#)" on page 423
- "[":PRINT](#)" on page 235
- "[":RCL \(Recall\)](#)" on page 191
- "[":SAV \(Save\)](#)" on page 195
- "[":VIEW](#)" on page 242

**Example Code**

```
' IMAGE_TRANSFER - In this example, we will query for the image data
' with ":DISPlay:DATA?", read the data, and then save it to a file.
Dim byteData() As Byte
myScope.IO.Timeout = 15000
myScope.WriteString ":DISPlay:DATA? BMP, COLOR"
byteData = myScope.ReadIEEEBlock(BinaryType_UI1)
' Output display data to a file:
strPath = "c:\scope\data\screen.bmp"
' Remove file if it exists.
If Len(Dir(strPath)) Then
    Kill strPath
End If
Close #1      ' If #1 is open, close it.
Open strPath For Binary Access Write Lock Write As #1      ' Open file for
or output.
Put #1, , byteData      ' Write data.
Close #1      ' Close file.
myScope.IO.Timeout = 5000
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

## :DISPlay:INTensity:WAveform

**N** (see [page 1334](#))

**Command Syntax** :DISPlay:INTensity:WAveform <value>

<value> ::= an integer from 0 to 100 in NR1 format.

The :DISPlay:INTensity:WAveform command sets the waveform intensity.

This is the same as adjusting the front panel [**Intensity**] knob.

**Query Syntax** :DISPlay:INTensity:WAveform?

The :DISPlay:INTensity:WAveform? query returns the waveform intensity setting.

**Return Format** <value><NL>

<value> ::= an integer from 0 to 100 in NR1 format.

**See Also** • "Introduction to :DISPlay Commands" on page 330

**:DISPlay:LABel****N** (see [page 1334](#))**Command Syntax**    `:DISPlay:LABel <value>``<value> ::= {{1 | ON} | {0 | OFF}}`

The :DISPlay:LABel command turns the analog and digital channel labels on and off.

**Query Syntax**    `:DISPlay:LABel?`

The :DISPlay:LABel? query returns the display mode of the analog and digital labels.

**Return Format**    `<value><NL>``<value> ::= {0 | 1}`

- See Also**
- "Introduction to [:DISPlay Commands](#)" on page 330
  - "[:CHANnel<n>:LABEL](#)" on page 287

**Example Code**

```
' DISP_LABEL  
' - Turns label names ON or OFF on the analyzer display.  
myScope.WriteString ":DISPlay:LABEL ON" ' Turn on labels.
```

See complete example programs at: [Chapter 42](#), "Programming Examples," starting on page 1343

**:DISPlay:LABList**

**N** (see [page 1334](#))

**Command Syntax**    `:DISPlay:LABList <binary block data>`  
`<binary block> ::= an ordered list of up to 75 labels, a maximum of 10 characters each, separated by newline characters.`

The :DISPlay:LABList command adds labels to the label list. Labels are added in alphabetical order.

**NOTE**

Labels that begin with the same alphabetic base string followed by decimal digits are considered duplicate labels. Duplicate labels are not added to the label list. For example, if label "A0" is in the list and you try to add a new label called "A123456789", the new label is not added.

**Query Syntax**    `:DISPlay:LABList?`

The :DISPlay:LABList? query returns the label list.

**Return Format**    `<binary block><NL>`  
`<binary block> ::= an ordered list of up to 75 labels, a maximum of 10 characters each, separated by newline characters.`

**See Also**

- "[Introduction to :DISPlay Commands](#)" on page 330
- "[":DISPlay:LABEL](#)" on page 341
- "[":CHANnel<n>:LABEL](#)" on page 287
- "[":DIGital<d>:LABEL](#)" on page 324
- "[":BUS<n>:LABEL](#)" on page 264

## :DISPlay:MENU

**N** (see [page 1334](#))

**Command Syntax** :DISPlay:MENU <menu>

<menu> ::= {MASK | MEASure | SEGmented | LISTer | POWer}

The :DISPlay:MENU command changes the front panel softkey menu.

## :DISPlay:SIDebar

**N** (see [page 1334](#))

**Command Syntax**    `:DISPlay:SIDebar <sidebar>`

```
<sidebar> ::= {SUMMary | CURSors | MEASurements | DVM | NAVigate  
| CONTrols | EVENTS | COUNTER}
```

The :DISPlay:SIDebar command specifies the sidebar dialog to display on the screen.

## :DISPlay:PERSistence

**N** (see [page 1334](#))

**Command Syntax** :DISPlay:PERSistence <value>

<value> ::= {MINimum | INFinite | <time>}

<time> ::= seconds in NR3 format from 100E-3 to 60E0

The :DISPlay:PERSistence command specifies the persistence setting:

- MINimum – indicates zero persistence.
- INFinite – indicates infinite persistence.
- <time> – for variable persistence, that is, you can specify how long acquisitions remain on the screen.

Use the :DISPlay:CLEar command to erase points stored by persistence.

**Query Syntax** :DISPlay:PERSistence?

The :DISPlay:PERSistence? query returns the specified persistence value.

**Return Format** <value><NL>

<value> ::= {MIN | INF | <time>}

**See Also**

- "[Introduction to :DISPlay Commands](#)" on page 330
- "[":DISPlay:CLEar](#)" on page 338

## :DISPlay:VECTors

**N** (see [page 1334](#))

**Command Syntax**    `:DISPlay:VECTors <vectors>`  
                      `<vectors> ::= {1 | ON}`

Vector display is always ON in the 3000T X-Series oscilloscopes.

When vectors are turned on, the oscilloscope displays lines connecting sampled data points.

**Query Syntax**    `:DISPlay:VECTors?`  
The :DISPlay:VECTors? query returns the vectors setting.

**Return Format**    `<vectors><NL>`  
                      `<vectors> ::= 1`

**See Also**    • "Introduction to :DISPlay Commands" on page 330

# 15 :DVM Commands

When the optional DSOXDVM digital voltmeter analysis feature is licensed, these commands control the digital voltmeter (DVM) feature.

**Table 96** :DVM Commands Summary

Command	Query	Options and Query Returns
:DVM:ARAnge {{0   OFF}   {1   ON}} (see page 348)	:DVM:ARAnge? (see page 348)	{0   1}
n/a	:DVM:CURREnt? (see page 349)	<dvm_value> ::= floating-point number in NR3 format
:DVM:ENABLE {{0   OFF}   {1   ON}} (see page 350)	:DVM:ENABLE? (see page 350)	{0   1}
:DVM:MODE <mode> (see page 351)	:DVM:MODE? (see page 351)	<dvm_mode> ::= {ACRMs   DC   DCRMs}
:DVM:SOURce <source> (see page 352)	:DVM:SOURce? (see page 352)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format

## :DVM:ARAnge

**N** (see [page 1334](#))

**Command Syntax**    `:DVM:ARAnge <setting>`

`<setting> ::= {{OFF | 0} | {ON | 1}}`

If the selected digital voltmeter (DVM) source channel is not used in oscilloscope triggering, the :DVM:ARAnge command turns the digital voltmeter's Auto Range capability on or off.

- When on, the DVM channel's vertical scale, vertical (ground level) position, and trigger (threshold voltage) level (used for the counter frequency measurement) are automatically adjusted.

The Auto Range capability overrides attempted adjustments of the channel's vertical scale and position.

- When off, you can adjust the channel's vertical scale and position normally.

**Query Syntax**    `:DVM:ARAnge?`

The :DVM:ARAnge? query returns a flag indicating whether the digital voltmeter's Auto Range capability is on or off.

**Return Format**    `<setting><NL>`

`<setting> ::= {0 | 1}`

**See Also**

- "[:DVM:SOURce](#)" on page 352
- "[:DVM:ENABLE](#)" on page 350
- "[:DVM:MODE](#)" on page 351

## :DVM:CURREnt

**N** (see [page 1334](#))

**Query Syntax** :DVM:CURREnt?

The :DVM:CURREnt? query returns the displayed 3-digit DVM value based on the current mode.

**Return Format** <dvm\_value><NL>

<dvm\_value> ::= floating-point number in NR3 format

**See Also**

- [":DVM:SOURce"](#) on page 352
- [":DVM:ENABLE"](#) on page 350
- [":DVM:MODE"](#) on page 351

**:DVM:ENABLE****N** (see [page 1334](#))**Command Syntax**    `:DVM:ENABLE <setting>``<setting> ::= {{OFF | 0} | {ON | 1}}`

The :DVM:ENABLE command turns the digital voltmeter (DVM) analysis feature on or off.

**Query Syntax**    `:DVM:ENABLE?`

The :DVM:ENABLE? query returns a flag indicating whether the digital voltmeter (DVM) analysis feature is on or off.

**Return Format**    `<setting><NL>``<setting> ::= {0 | 1}`**See Also**

- [":DVM:SOURce" on page 352](#)
- [":DVM:MODE" on page 351](#)
- [":DVM:ARAnge" on page 348](#)

**:DVM:MODE**

**N** (see [page 1334](#))

**Command Syntax**    `:DVM:MODE <dvm_mode>`

`<dvm_mode> ::= {ACRMs | DC | DCRMs}`

The :DVM:MODE command sets the digital voltmeter (DVM) mode:

- ACRMs – displays the root-mean-square value of the acquired data, with the DC component removed.
- DC – displays the DC value of the acquired data.
- DCRMs – displays the root-mean-square value of the acquired data.

**Query Syntax**    `:DVM:MODE?`

The :DVM:MODE? query returns the selected DVM mode.

**Return Format**    `<dvm_mode><NL>`

`<dvm_mode> ::= {ACRM | DC | DCRM}`

**See Also**    [":DVM:ENABLE" on page 350](#)  
[":DVM:SOURce" on page 352](#)  
[":DVM:ARAnge" on page 348](#)  
[":DVM:CURREnt" on page 349](#)

**:DVM:SOURce****N** (see [page 1334](#))**Command Syntax**    `:DVM:SOURce <source>`    `<source> ::= {CHANnel<n>}`    `<n> ::= 1-2 or 1-4 in NR1 format`

The :DVM:SOURce command sets the select the analog channel on which digital voltmeter (DVM) measurements are made.

The selected channel does not have to be on (displaying a waveform) in order for DVM measurements to be made.

**Query Syntax**    `:DVM:SOURce?`

The :DVM:SOURce? query returns the selected DVM input source.

**Return Format**    `<source><NL>`    `<source> ::= {CHAN<n>}`    `<n> ::= 1-2 or 1-4 in NR1 format`**See Also**

- [":DVM:ENABLE"](#) on page 350
- [":DVM:MODE"](#) on page 351
- [":DVM:ARAnge"](#) on page 348
- [":DVM:CURRent"](#) on page 349

# 16 :EXTernal Trigger Commands

Control the input characteristics of the external trigger input. See "[Introduction to :EXTernal Trigger Commands](#)" on page 353.

**Table 97** :EXTernal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXTernal:BWLIMIT <bwlimit> (see <a href="#">page 354</a> )	:EXTernal:BWLIMIT? (see <a href="#">page 354</a> )	<bwlimit> ::= {0   OFF}
:EXTernal:PROBE <attenuation> (see <a href="#">page 355</a> )	:EXTernal:PROBE? (see <a href="#">page 355</a> )	<attenuation> ::= probe attenuation ratio in NR3 format
:EXTernal:RANGE <range>[<suffix>] (see <a href="#">page 356</a> )	:EXTernal:RANGE? (see <a href="#">page 356</a> )	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V   mV}
:EXTernal:UNITS <units> (see <a href="#">page 357</a> )	:EXTernal:UNITS? (see <a href="#">page 357</a> )	<units> ::= {VOLT   AMPere}

**Introduction to :EXTernal Trigger Commands** The EXTernal trigger subsystem commands control the input characteristics of the external trigger input. The probe factor, impedance, input range, input protection state, units, and bandwidth limit settings may all be queried. Depending on the instrument type, some settings may be changeable.

## Reporting the Setup

Use :EXTernal? to query setup information for the EXTernal subsystem.

## Return Format

The following is a sample response from the :EXTernal query. In this case, the query was issued following a \*RST command.

```
:EXT:BWL 0;RANG +8.0E+00;UNIT VOLT;PROB +1.000E+00
```

**:EXTernal:BWLimits**(see [page 1334](#))

**Command Syntax**    `:EXTernal:BWLimits <bwlimits>`  
                      `<bwlimits> ::= {0 | OFF}`

The :EXTernal:BWLimits command is provided for product compatibility. The only legal value is 0 or OFF. Use the :TRIGger:HFReject command to limit bandwidth on the external trigger input.

**Query Syntax**    `:EXTernal:BWLimits?`

The :EXTernal:BWLimits? query returns the current setting of the low-pass filter (always 0).

**Return Format**    `<bwlimits><NL>`  
                      `<bwlimits> ::= 0`

**See Also**

- ["Introduction to :EXTernal Trigger Commands" on page 353](#)
- ["Introduction to :TRIGger Commands" on page 1047](#)
- [":TRIGger:HFReject" on page 1051](#)

## :EXTernal:PROBe



(see [page 1334](#))

### Command Syntax

```
:EXTernal:PROBe <attenuation>
<attenuation> ::= probe attenuation ratio in NR3 format
```

The :EXTernal:PROBe command specifies the probe attenuation factor for the external trigger. The probe attenuation factor may be 0.1 to 1000. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors and for setting trigger levels.

If an AutoProbe probe is connected to the oscilloscope, the attenuation value cannot be changed from the sensed value. Attempting to set the oscilloscope to an attenuation value other than the sensed value produces an error.

### Query Syntax

```
:EXTernal:PROBe?
```

The :EXTernal:PROBe? query returns the current probe attenuation factor for the external trigger.

### Return Format

```
<attenuation><NL>
<attenuation> ::= probe attenuation ratio in NR3 format
```

### See Also

- ["Introduction to :EXTernal Trigger Commands" on page 353](#)
- [":EXTernal:RANGE" on page 356](#)
- ["Introduction to :TRIGger Commands" on page 1047](#)
- [":CHANnel<n>:PROBe" on page 289](#)

**:EXTernal:RANGE**(see [page 1334](#))**Command Syntax**    `:EXTernal:RANGE <range>[<suffix>]`    `<range> ::= vertical full-scale range value in NR3 format`    `<suffix> ::= {V | mV}`

The :EXTernal:RANGE command is provided for product compatibility. When using 1:1 probe attenuation, the range can only be set to 8.0 V.

If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

**Query Syntax**    `:EXTernal:RANGE?`

The :EXTernal:RANGE? query returns the current full-scale range setting for the external trigger.

**Return Format**    `<range_argument><NL>`    `<range_argument> ::= external trigger range value in NR3 format`**See Also**

- "[Introduction to :EXTernal Trigger Commands](#)" on page 353

- "[:EXTernal:PROBe](#)" on page 355

- "[Introduction to :TRIGger Commands](#)" on page 1047

## :EXTernal:UNITS

**N** (see [page 1334](#))

**Command Syntax**    `:EXTernal:UNITS <units>`  
`<units> ::= {VOLT | AMPere}`

The :EXTernal:UNITS command sets the measurement units for the probe connected to the external trigger input. Select VOLT for a voltage probe and select AMPere for a current probe. Measurement results, channel sensitivity, and trigger level will reflect the measurement units you select.

**Query Syntax**    `:EXTernal:UNITS?`

The :CHANnel<n>:UNITS? query returns the current units setting for the external trigger.

**Return Format**    `<units><NL>`  
`<units> ::= {VOLT | AMP}`

**See Also**

- "[Introduction to :EXTernal Trigger Commands](#)" on page 353
- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":EXTernal:RANGE](#)" on page 356
- "[":EXTernal:PROBe](#)" on page 355
- "[":CHANnel<n>:UNITS](#)" on page 298



# 17 :FFT Commands

Control the FFT function in the oscilloscope. See "[Introduction to :FFT Commands](#)" on page 360.

**Table 98** :FFT Commands Summary

Command	Query	Options and Query Returns
:FFT:AVERage:COUNT <count> (see <a href="#">page 361</a> )	:FFT:AVERage:COUNT? (see <a href="#">page 361</a> )	<count> ::= an integer from 2 to 65536 in NR1 format.
:FFT:CENTER <frequency> (see <a href="#">page 362</a> )	:FFT:CENTER? (see <a href="#">page 362</a> )	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.
:FFT:CLEar (see <a href="#">page 363</a> )	n/a	n/a
:FFT:DISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 364</a> )	:FFT:DISPLAY? (see <a href="#">page 364</a> )	<s> ::= 1-6, in NR1 format. {0   1}
:FFT:DMODE <display_mode> (see <a href="#">page 365</a> )	:FFT:DMODE? (see <a href="#">page 365</a> )	<display_mode> ::= {NORMal   AVERage   MAXHold   MINHold}
:FFT:FREQuency:STARt <frequency> (see <a href="#">page 367</a> )	:FFT:FREQuency:STARt? (see <a href="#">page 367</a> )	<frequency> ::= the start frequency in NR3 format.
:FFT:FREQuency:STOP <frequency> (see <a href="#">page 368</a> )	:FFT:FREQuency:STOP? (see <a href="#">page 368</a> )	<frequency> ::= the stop frequency in NR3 format.
:FFT:GATE <gating> (see <a href="#">page 369</a> )	:FFT:GATE? (see <a href="#">page 369</a> )	<gating> ::= {NONE   ZOOM}
:FFT:OFFSet <offset> (see <a href="#">page 370</a> )	:FFT:OFFSet? (see <a href="#">page 370</a> )	<offset> ::= the value at center screen in NR3 format.
:FFT:RANGE <range> (see <a href="#">page 371</a> )	:FFT:RANGE? (see <a href="#">page 371</a> )	<range> ::= the full-scale vertical axis value in NR3 format.

**Table 98** :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFT:REFerence <level> (see <a href="#">page 372</a> )	:FFT:REFerence? (see <a href="#">page 372</a> )	<level> ::= the current reference level in NR3 format.
:FFT:SCALe <scale value>[<suffix>] (see <a href="#">page 373</a> )	:FFT:SCALe? (see <a href="#">page 373</a> )	<scale_value> ::= integer in NR1 format. <suffix> ::= dB
:FFT:SOURcel <source> (see <a href="#">page 374</a> )	:FFT:SOURcel? (see <a href="#">page 374</a> )	<source> ::= {CHANnel<n>   FUNCtion<c>   MATH<c>} <n> ::= 1 to (# analog channels) in NR1 format. <c> ::= {1   2}
:FFT:SPAN <span> (see <a href="#">page 375</a> )	:FFT:SPAN? (see <a href="#">page 375</a> )	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.
:FFT:VTYPe <units> (see <a href="#">page 376</a> )	:FFT:VTYPe? (see <a href="#">page 376</a> )	<units> ::= {DECibel   VRMS}
:FFT:WINDOW <window> (see <a href="#">page 377</a> )	:FFT:WINDOW? (see <a href="#">page 377</a> )	<window> ::= {RECTangular   HANNing   FLATtop   BHARRis}

**Introduction to :FFT Commands** The FFT subsystem controls the FFT function in the oscilloscope.

#### Reporting the Setup

Use :FFT? to query setup information for the FFT subsystem.

#### Return Format

The following is a sample response from the :FFT? query. In this case, the query was issued following a \*RST command.

```
:FFT:DISP 0;SOUR1 CHAN1;RANG +160E+00;OFFS -60.0000E+00;SPAN
+100.0000E+03;CENT +50.000000E+03;WIND HANN;VTYP DEC;DMODE
NORM;AVER:COUN 8
```

## :FFT:AVERage:COUNt

**N** (see [page 1334](#))

**Command Syntax** :FFT:AVERage:COUNt <count>

<count> ::= an integer from 2 to 65536 in NR1 format.

The :FFT:AVERage:COUNt command sets the number of waveforms to be averaged together.

The number of averages can be set from 2 to 65536 in increments of powers of 2.

Increasing the number of averages will increase resolution and reduce noise.

**Query Syntax** :FFT:AVERage:COUNt?

The :FFT:AVERage:COUNt? query returns the number of waveforms to be averaged together.

**Return Format** <count><NL>

<count> ::= an integer from 2 to 65536 in NR1 format.

**See Also**

- "[:FFT:CENTer](#)" on page 362
- "[:FFT:CLEar](#)" on page 363
- "[:FFT:DISPlay](#)" on page 364
- "[:FFT:DMDe](#)" on page 365
- "[:FFT:OFFSet](#)" on page 370
- "[:FFT:RANGE](#)" on page 371
- "[:FFT:REFerence](#)" on page 372
- "[:FFT:SCALe](#)" on page 373
- "[:FFT:SOURce1](#)" on page 374
- "[:FFT:SPAN](#)" on page 375
- "[:FFT:FREQuency:STARt](#)" on page 367
- "[:FFT:FREQuency:STOP](#)" on page 368
- "[:FFT:VTPe](#)" on page 376
- "[:FFT:WINDOW](#)" on page 377

**:FFT:CENTer**

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:CENTer <frequency>`

`<frequency>` ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.

The :FFT:CENTer command sets the center frequency when FFT (Fast Fourier Transform) is selected.

**Query Syntax**    `:FFT:CENTer?`

The :FFT:CENTer? query returns the current center frequency in Hertz.

**Return Format**    `<frequency><NL>`

`<frequency>` ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.

**NOTE**

After a \*RST (Reset) or :AUToscale command, the values returned by the :FFT:CENTer? and :FFT:SPAN? queries depend on the current :TIMEbase:RANGE value. Once you change either the :FFT:CENTer or :FFT:SPAN value, they no longer track the :TIMEbase:RANGE value.

**See Also**

- "[:FFT:AVERage:COUNT](#)" on page 361
- "[:FFT:CENTER](#)" on page 362
- "[:FFT:DISPLAY](#)" on page 364
- "[:FFT:DMDMode](#)" on page 365
- "[:FFT:OFFSet](#)" on page 370
- "[:FFT:RANGE](#)" on page 371
- "[:FFT:REFERENCE](#)" on page 372
- "[:FFT:SCALE](#)" on page 373
- "[:FFT:SOURce1](#)" on page 374
- "[:FFT:SPAN](#)" on page 375
- "[:FFT:FREQuency:STARt](#)" on page 367
- "[:FFT:FREQuency:STOP](#)" on page 368
- "[:FFT:VTPe](#)" on page 376
- "[:FFT:WINDOW](#)" on page 377

## :FFT:CLEar

**N** (see [page 1334](#))

**Command Syntax** :FFT:CLEar

When the FFT display mode is AVERage, MAXHold, or MINHold, the :FFT:CLEar command clears the number of evaluated waveforms.

**See Also**

- "[:FFT:AVERage:COUNT](#)" on page 361
- "[:FFT:CENTer](#)" on page 362
- "[:FFT:DISPlay](#)" on page 364
- "[:FFT:DModE](#)" on page 365
- "[:FFT:OFFSet](#)" on page 370
- "[:FFT:RANGe](#)" on page 371
- "[:FFT:REFerence](#)" on page 372
- "[:FFT:SCALe](#)" on page 373
- "[:FFT:SOURce1](#)" on page 374
- "[:FFT:SPAN](#)" on page 375
- "[:FFT:FREQuency:STARt](#)" on page 367
- "[:FFT:FREQuency:STOP](#)" on page 368
- "[:FFT:VTPe](#)" on page 376
- "[:FFT:WINDOW](#)" on page 377

**:FFT:DISPlay**

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:DISPLAY {{0 | OFF} | {1 | ON}}`

The :FFT:DISPLAY command turns the display of the FFT function on or off.

When ON is selected, the FFT function is calculated and displayed.

When OFF is selected, the FFT function is neither calculated nor displayed.

**Query Syntax**    `:FFT:DISPLAY?`

The :FFT:DISPLAY? query returns whether the function display is on or off.

**Return Format**    `<display><NL>`

`<display> ::= {0 | 1}`

**See Also**    [":FFT:AVERage:COUNt"](#) on page 361

[":FFT:CENTer"](#) on page 362

[":FFT:CLEar"](#) on page 363

[":FFT:DMODe"](#) on page 365

[":FFT:OFFSet"](#) on page 370

[":FFT:RANGe"](#) on page 371

[":FFT:REFerence"](#) on page 372

[":FFT:SCALe"](#) on page 373

[":FFT:SOURce1"](#) on page 374

[":FFT:SPAN"](#) on page 375

[":FFT:FREQuency:STARt"](#) on page 367

[":FFT:FREQuency:STOP"](#) on page 368

[":FFT:VTPe"](#) on page 376

[":FFT:WINDOW"](#) on page 377

## :FFT:DMODE

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:FFT:DMODE &lt;display_mode&gt;</code> <code>&lt;display_mode&gt; ::= {NORMal   AVERage   MAXHold   MINHold}</code>
	The :FFT:DMODE command selects one of the FFT waveform display modes:
	<ul style="list-style-type: none"> <li>· NORMal – this is the FFT waveform without any averaging or hold functions applied. This is how FFT math function waveforms are displayed.</li> <li>· AVERage – the FFT waveform is averaged the selected number of times. Averages are calculated using a "decaying average" approximation, where:  <math display="block">\text{next\_average} = \text{current\_average} + (\text{new\_data} - \text{current\_average})/N</math> Where N starts at 1 for the first acquisition and increments for each following acquisition until it reaches the selected number of averages, where it holds.</li> <li>· MAXHold – records the maximum vertical values found at each horizontal bucket across multiple analysis cycles and uses those values to build a waveform. This display mode is often referred to as Max Envelope.</li> <li>· MINHold – records the minimum vertical values found at each horizontal bucket across multiple analysis cycles and uses those values to build a waveform. This display mode is often referred to as Min Envelope.</li> </ul>
<b>Query Syntax</b>	<code>:FFT:DMODE?</code>
	The :FFT:DMODE? query returns the currently set FFT display mode.
<b>Return Format</b>	<code>&lt;display_mode&gt;&lt;NL&gt;</code> <code>&lt;display_mode&gt; ::= {NORM   AVER   MAXH   MINH}</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">:FFT:AVERage:COUNT</a>" on page 361</li> <li>· "<a href="#">:FFT:CENTER</a>" on page 362</li> <li>· "<a href="#">:FFT:CLEar</a>" on page 363</li> <li>· "<a href="#">:FFT:DISPLAY</a>" on page 364</li> <li>· "<a href="#">:FFT:OFFSet</a>" on page 370</li> <li>· "<a href="#">:FFT:RANGE</a>" on page 371</li> <li>· "<a href="#">:FFT:REFERENCE</a>" on page 372</li> <li>· "<a href="#">:FFT:SCALE</a>" on page 373</li> <li>· "<a href="#">:FFT:SOURce1</a>" on page 374</li> <li>· "<a href="#">:FFT:SPAN</a>" on page 375</li> <li>· "<a href="#">:FFT:FREQuency:START</a>" on page 367</li> <li>· "<a href="#">:FFT:FREQuency:STOP</a>" on page 368</li> <li>· "<a href="#">:FFT:VTPe</a>" on page 376</li> </ul>

- [":FFT:WINDow"](#) on page 377

## :FFT:FREQuency:STARt

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:FFT:FREQuency:STARt &lt;frequency&gt;</code> <code>&lt;frequency&gt; ::= the start frequency in NR3 format.</code>
	The :FFT:FREQuency:STARt command sets the start frequency in the FFT (Fast Fourier Transform) math function's displayed range.
	The FFT (Fast Fourier Transform) math function's displayed range can also be set with the :FFT:CENTER and :FFT:SPAN commands.
<b>Query Syntax</b>	<code>:FFT:FREQuency:STARt?</code>
	The :FFT:FREQuency:STARt? query returns the current start frequency in Hertz.
<b>Return Format</b>	<code>&lt;frequency&gt;&lt;NL&gt;</code> <code>&lt;frequency&gt; ::= the start frequency in NR3 format.</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">":FFT:AVERage:COUNT"</a> on page 361</li> <li>· <a href="#">":FFT:CENTER"</a> on page 362</li> <li>· <a href="#">":FFT:CLEar"</a> on page 363</li> <li>· <a href="#">":FFT:DISPLAY"</a> on page 364</li> <li>· <a href="#">":FFT:DMDOf"</a> on page 365</li> <li>· <a href="#">":FFT:OFFSet"</a> on page 370</li> <li>· <a href="#">":FFT:RANGE"</a> on page 371</li> <li>· <a href="#">":FFT:REFERENCE"</a> on page 372</li> <li>· <a href="#">":FFT:SCALE"</a> on page 373</li> <li>· <a href="#">":FFT:SOURce1"</a> on page 374</li> <li>· <a href="#">":FFT:SPAN"</a> on page 375</li> <li>· <a href="#">":FFT:FREQuency:STOP"</a> on page 368</li> <li>· <a href="#">":FFT:VTPe"</a> on page 376</li> <li>· <a href="#">":FFT:WINDOW"</a> on page 377</li> </ul>

## :FFT:FREQuency:STOP

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:FREQuency:STOP <frequency>`

`<frequency>` ::= the stop frequency in NR3 format.

The :FFT:FREQuency:STOP command sets the stop frequency in the FFT (Fast Fourier Transform) math function's displayed range.

The FFT (Fast Fourier Transform) math function's displayed range can also be set with the :FFT:CENTER and :FFT:SPAN commands.

**Query Syntax**    `:FFT:FREQuency:STOP?`

The :FFT:FREQuency:STOP? query returns the current stop frequency in Hertz.

**Return Format**    `<frequency><NL>`

`<frequency>` ::= the stop frequency in NR3 format.

**See Also**

- [":FFT:AVERage:COUNt"](#) on page 361
- [":FFT:CENTER"](#) on page 362
- [":FFT:CLEar"](#) on page 363
- [":FFT:DISPlay"](#) on page 364
- [":FFT:DMDe"](#) on page 365
- [":FFT:OFFSet"](#) on page 370
- [":FFT:RANGE"](#) on page 371
- [":FFT:REFerence"](#) on page 372
- [":FFT:SCALe"](#) on page 373
- [":FFT:SOURce1"](#) on page 374
- [":FFT:SPAN"](#) on page 375
- [":FFT:FREQuency:START"](#) on page 367
- [":FFT:VTPe"](#) on page 376
- [":FFT:WINDOW"](#) on page 377

**:FFT:GATE**

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:GATE <gating>`  
                      `<gating> ::= {NONE | ZOOM}`

The :FFT:GATE command specifies whether the FFT is performed on the Main time base window (NONE) or the ZOOM window when the zoomed time base is displayed.

**Query Syntax**    `:FFT:GATE?`  
The :FFT:GATE? query returns the gate setting.

**Return Format**    `<gating><NL>`  
                      `<gating> ::= {NONE | ZOOM}`

**See Also**

- [":FFT:VTPe"](#) on page 376
- [":FFT:WINDOW"](#) on page 377
- ["Introduction to :FFT Commands"](#) on page 360

## :FFT:OFFSet

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:OFFSet <offset>`

`<offset>` ::= the value at center screen in NR3 format.

The :FFT:OFFSet command specifies the FFT vertical value represented at center screen.

If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value.

### NOTE

The :FFT:OFFSet command is equivalent to the :FFT:REFerence command.

**Query Syntax**    `:FFT:OFFSet?`

The :FFT:OFFSet? query returns the current offset value for the FFT function.

**Return Format**    `<offset><NL>`

`<offset>` ::= the value at center screen in NR3 format.

- See Also**
- [":FFT:AVERage:COUNt"](#) on page 361
  - [":FFT:CENTer"](#) on page 362
  - [":FFT:CLEar"](#) on page 363
  - [":FFT:DISPlay"](#) on page 364
  - [":FFT:DMODe"](#) on page 365
  - [":FFT:RANGe"](#) on page 371
  - [":FFT:REFerence"](#) on page 372
  - [":FFT:SCALe"](#) on page 373
  - [":FFT:SOURce1"](#) on page 374
  - [":FFT:SPAN"](#) on page 375
  - [":FFT:FREQuency:STARt"](#) on page 367
  - [":FFT:FREQuency:STOP"](#) on page 368
  - [":FFT:VTPe"](#) on page 376
  - [":FFT:WINDOW"](#) on page 377

**:FFT:RANGE**

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:RANGE <range>`

`<range>` ::= the full-scale vertical axis value in NR3 format.

The :FFT:RANGE command defines the full-scale vertical axis for the FFT function.

**Query Syntax**    `:FFT:RANGE?`

The :FFT:RANGE? query returns the current full-scale range value for the FFT function.

**Return Format**    `<range><NL>`

`<range>` ::= the full-scale vertical axis value in NR3 format.

- See Also**
- "[:FFT:AVERage:COUNT](#)" on page 361
  - "[:FFT:CENTER](#)" on page 362
  - "[:FFT:CLEar](#)" on page 363
  - "[:FFT:DISPLAY](#)" on page 364
  - "[:FFT:D MODE](#)" on page 365
  - "[:FFT:OFFSet](#)" on page 370
  - "[:FFT:REFerence](#)" on page 372
  - "[:FFT:SCALe](#)" on page 373
  - "[:FFT:SOURce1](#)" on page 374
  - "[:FFT:SPAN](#)" on page 375
  - "[:FFT:FREQuency:STARt](#)" on page 367
  - "[:FFT:FREQuency:STOP](#)" on page 368
  - "[:FFT:VTPe](#)" on page 376
  - "[:FFT:WINDOW](#)" on page 377

## :FFT:REFerence

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:REFerence <level>`

`<level>` ::= the current reference level in NR3 format.

The :FFT:REFerence command specifies the FFT vertical value represented at center screen.

If you set the reference level to a value outside of the legal range, the level is automatically set to the nearest legal value.

### NOTE

The :FFT:REFerence command is equivalent to the :FFT:OFFSet command.

**Query Syntax**    `:FFT:REFerence?`

The :FFT:REFerence? query returns the current reference level value for the FFT function.

**Return Format**    `<level><NL>`

`<level>` ::= the current reference level in NR3 format.

- See Also**
- "[:FFT:AVERage:COUNT](#)" on page 361
  - "[:FFT:CENTER](#)" on page 362
  - "[:FFT:CLEar](#)" on page 363
  - "[:FFT:DISPlay](#)" on page 364
  - "[:FFT:DMode](#)" on page 365
  - "[:FFT:OFFSet](#)" on page 370
  - "[:FFT:RANGE](#)" on page 371
  - "[:FFT:SCALe](#)" on page 373
  - "[:FFT:SOURce1](#)" on page 374
  - "[:FFT:SPAN](#)" on page 375
  - "[:FFT:FREQuency:STARt](#)" on page 367
  - "[:FFT:FREQuency:STOP](#)" on page 368
  - "[:FFT:VTPe](#)" on page 376
  - "[:FFT:WINDOW](#)" on page 377

**:FFT:SCALe**

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:SCALe <scale_value>[<suffix>]`

`<scale_value>` ::= floating-point value in NR3 format.

`<suffix>` ::= dB

The :FFT:SCALe command sets the vertical scale, or units per division, of the FFT function. Legal values for the scale depend on the selected function.

**Query Syntax**    `:FFT:SCALe?`

The :FFT:SCALe? query returns the current scale value for the FFT function.

**Return Format**    `<scale_value><NL>`

`<scale_value>` ::= floating-point value in NR3 format.

- See Also**
- "[:FFT:AVERage:COUNt](#)" on page 361
  - "[:FFT:CENTER](#)" on page 362
  - "[:FFT:CLEar](#)" on page 363
  - "[:FFT:DISPlay](#)" on page 364
  - "[:FFT:DMD](#)" on page 365
  - "[:FFT:OFFSet](#)" on page 370
  - "[:FFT:RANGe](#)" on page 371
  - "[:FFT:REFerence](#)" on page 372
  - "[:FFT:SOURce1](#)" on page 374
  - "[:FFT:SPAN](#)" on page 375
  - "[:FFT:FREQuency:STARt](#)" on page 367
  - "[:FFT:FREQuency:STOP](#)" on page 368
  - "[:FFT:VTPe](#)" on page 376
  - "[:FFT:WINDOW](#)" on page 377

## :FFT:SOURce1

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:SOURce1 <offset>`

```
<source> ::= {CHANnel<n> | FUNCtion<c> | MATH<c>}
<n> ::= 1 to (# analog channels) in NR1 format.
<c> ::= {1 | 2}
```

The :FFT:SOURce1 command selects the source for the FFT function.

### NOTE

Another shorthand notation for SOURce1 in this command/query (besides SOUR1) is SOUR.

**Query Syntax**    `:FFT:SOURce1?`

The :FFT:SOURce1? query returns the current source1 for the FFT function.

**Return Format**    `<source><NL>`

```
<source> ::= {CHANnel<n> | FUNCtion<c> | MATH<c>}
<n> ::= 1 to (# analog channels) in NR1 format.
<c> ::= {1 | 2}
```

**See Also**

- [":FFT:AVERage:COUNT"](#) on page 361
- [":FFT:CENTER"](#) on page 362
- [":FFT:CLEar"](#) on page 363
- [":FFT:DISPLAY"](#) on page 364
- [":FFT:D MODE"](#) on page 365
- [":FFT:OFFSet"](#) on page 370
- [":FFT:RANGE"](#) on page 371
- [":FFT:REFERENCE"](#) on page 372
- [":FFT:SCALE"](#) on page 373
- [":FFT:SPAN"](#) on page 375
- [":FFT:FREQuency:STARt"](#) on page 367
- [":FFT:FREQuency:STOP"](#) on page 368
- [":FFT:VTPe"](#) on page 376
- [":FFT:WINDOW"](#) on page 377

## :FFT:SPAN

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:SPAN <span>`

`<span>` ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

If you set the frequency span to a value outside of the legal range, the step size is automatically set to the nearest legal value.

The :FFT:SPAN command sets the frequency span of the display (left graticule to right graticule) when FFT (Fast Fourier Transform) is selected.

**Query Syntax**    `:FFT:SPAN?`

The :FFT:SPAN? query returns the current frequency span in Hertz.

### NOTE

After a \*RST (Reset) or :AUToscale command, the values returned by the :FFT:CENTER? and :FFT:SPAN? queries depend on the current :TIMEbase:RANGE value. Once you change either the :FFT:CENTER or :FFT:SPAN value, they no longer track the :TIMEbase:RANGE value.

---

**Return Format**    `<span><NL>`

`<span>` ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

**See Also**

- "[:FFT:AVERage:COUNt](#)" on page 361
- "[:FFT:CENTER](#)" on page 362
- "[:FFT:CLEar](#)" on page 363
- "[:FFT:DISPlay](#)" on page 364
- "[:FFT:DMD](#)" on page 365
- "[:FFT:OFFSet](#)" on page 370
- "[:FFT:RANGE](#)" on page 371
- "[:FFT:REFerence](#)" on page 372
- "[:FFT:SCALe](#)" on page 373
- "[:FFT:SOURce1](#)" on page 374
- "[:FFT:FREQuency:STARt](#)" on page 367
- "[:FFT:FREQuency:STOP](#)" on page 368
- "[:FFT:VTPe](#)" on page 376
- "[:FFT:WINDow](#)" on page 377

**:FFT:VTYPe****N** (see [page 1334](#))**Command Syntax**    `:FFT:VTYPe <units>``<units> ::= {DECibel | VRMS}`

The :FFT:VTYPe command specifies FFT vertical units as DECibel or VRMS.

**Query Syntax**    `:FFT:VTYPe?`

The :FFT:VTYPe? query returns the current FFT vertical units.

**Return Format**    `<units><NL>``<units> ::= {DEC | VRMS}`**See Also**

- [":FFT:AVERage:COUNt"](#) on page 361

- [":FFT:CENTER"](#) on page 362

- [":FFT:CLEar"](#) on page 363

- [":FFT:DISPlay"](#) on page 364

- [":FFT:DMDe"](#) on page 365

- [":FFT:OFFSet"](#) on page 370

- [":FFT:RANGe"](#) on page 371

- [":FFT:REFerence"](#) on page 372

- [":FFT:SCALe"](#) on page 373

- [":FFT:SOURce1"](#) on page 374

- [":FFT:SPAN"](#) on page 375

- [":FFT:FREQuency:STARt"](#) on page 367

- [":FFT:FREQuency:STOP"](#) on page 368

- [":FFT:WINDOW"](#) on page 377

## :FFT:WINDOW

**N** (see [page 1334](#))

**Command Syntax**    `:FFT:WINDOW <window>`

$$<\text{window}> ::= \{\text{RECTangular} \mid \text{HANNing} \mid \text{FLATtop} \mid \text{BHARris}\}$$

The :FFT:WINDOW command allows the selection of four different windowing transforms or operations for the FFT (Fast Fourier Transform) function.

The FFT operation assumes that the time record repeats. Unless an integral number of sampled waveform cycles exist in the record, a discontinuity is created between the end of one record and the beginning of the next. This discontinuity introduces additional frequency components about the peaks into the spectrum. This is referred to as leakage. To minimize leakage, windows that approach zero smoothly at the start and end of the record are employed as filters to the FFTs. Each window is useful for certain classes of input signals.

- RECTangular – useful for transient signals, and signals where there are an integral number of cycles in the time record.
- HANNing – useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together, or for making frequency measurements. This is the default window.
- FLATtop – best for making accurate amplitude measurements of frequency peaks.
- BHARris (Blackman-Harris) – reduces time resolution compared to the rectangular window, but it improves the capacity to detect smaller impulses due to lower secondary lobes (provides minimal spectral leakage).

**Query Syntax**    `:FFT:WINDOW?`

The :FFT:WINDOW? query returns the value of the window selected for the FFT function.

**Return Format**    `<window><NL>`

$$<\text{window}> ::= \{\text{RECT} \mid \text{HANN} \mid \text{FLAT} \mid \text{BHAR}\}$$

**See Also**

- "[:FFT:AVERage:COUNt](#)" on page 361
- "[:FFT:CENTer](#)" on page 362
- "[:FFT:CLEar](#)" on page 363
- "[:FFT:DISPlay](#)" on page 364
- "[:FFT:DMDe](#)" on page 365
- "[:FFT:OFFSet](#)" on page 370
- "[:FFT:RANGe](#)" on page 371
- "[:FFT:REFerence](#)" on page 372
- "[:FFT:SCALe](#)" on page 373

- [":FFT:SOURce1" on page 374](#)
- [":FFT:SPAN" on page 375](#)
- [":FFT:FREQuency:STARt" on page 367](#)
- [":FFT:FREQuency:STOP" on page 368](#)
- [":FFT:VTPe" on page 376](#)

# 18 :FUNCTION<m> Commands

Control math functions in the oscilloscope. See "[Introduction to :FUNCTION<m> Commands](#)" on page 383.

**Table 99** :FUNCTION<m> Commands Summary

Command	Query	Options and Query Returns
:FUNCTION<m>:AVERage:COUNT <count> (see <a href="#">page 384</a> )	:FUNCTION<m>:AVERage:COUNT? (see <a href="#">page 384</a> )	<count> ::= an integer from 2 to 65536 in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:CLOCK <source> (see <a href="#">page 385</a> )	:FUNCTION<m>:BUS:CLOCK? (see <a href="#">page 385</a> )	<source> ::= {CHANnel<n>   DIGItal<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:SLOPE <slope> (see <a href="#">page 386</a> )	:FUNCTION<m>:BUS:SLOPE? (see <a href="#">page 386</a> )	<slope> ::= {NEGative   POSitive   EITHER} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YINCREMENT <value> (see <a href="#">page 387</a> )	:FUNCTION<m>:BUS:YINCREMENT? (see <a href="#">page 387</a> )	<value> ::= value per bus code, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YORIGIN <value> (see <a href="#">page 388</a> )	:FUNCTION<m>:BUS:YORIGIN? (see <a href="#">page 388</a> )	<value> ::= value at bus code = 0, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YUNITS <units> (see <a href="#">page 389</a> )	:FUNCTION<m>:BUS:YUNITS? (see <a href="#">page 389</a> )	<units> ::= {VOLT   AMPere   NONE} <m> ::= 1 to (# math functions) in NR1 format

**Table 99** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:CLEAR (see <a href="#">page 390</a> )	n/a	n/a
:FUNCTION<m>:DISPLAY { {0   OFF}   {1   ON}} (see <a href="#">page 391</a> )	:FUNCTION<m>:DISPLAY? (see <a href="#">page 391</a> )	{0   1} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:CENTer <frequency> (see <a href="#">page 392</a> )	:FUNCTION<m>[:FFT]:CENTer? (see <a href="#">page 392</a> )	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:FREQuency:STARt <frequency> (see <a href="#">page 393</a> )	:FUNCTION<m>[:FFT]:FREQuency:STARt? (see <a href="#">page 393</a> )	<frequency> ::= the start frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:FREQuency:STOP <frequency> (see <a href="#">page 394</a> )	:FUNCTION<m>[:FFT]:FREQuency:STOP? (see <a href="#">page 394</a> )	<frequency> ::= the stop frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:GATE <gating> (see <a href="#">page 395</a> )	:FUNCTION<m>[:FFT]:GATE? (see <a href="#">page 395</a> )	<gating> ::= {NONE   ZOOM} <m> ::= 1-4 in NR1 format
:FUNCTION<m>[:FFT]:SPAN <span> (see <a href="#">page 396</a> )	:FUNCTION<m>[:FFT]:SPAN? (see <a href="#">page 396</a> )	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:VTPe <units> (see <a href="#">page 397</a> )	:FUNCTION<m>[:FFT]:VTPe? (see <a href="#">page 397</a> )	<units> ::= {DECibel   VRMS} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:WINDow <>window> (see <a href="#">page 398</a> )	:FUNCTION<m>[:FFT]:WINDow? (see <a href="#">page 398</a> )	<>window> ::= {RECTangular   HANNing   FLATtop   BHARris} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:FREQuency:HIGHpass <3dB_freq> (see <a href="#">page 399</a> )	:FUNCTION<m>:FREQuency:HIGHpass? (see <a href="#">page 399</a> )	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format

**Table 99** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:FREQuenc y:LOWPass <3dB_freq> (see <a href="#">page 400</a> )	:FUNCTION<m>:FREQuenc y:LOWPass? (see <a href="#">page 400</a> )	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:INTegrat e:IOFFset <input_offset> (see <a href="#">page 401</a> )	:FUNCTION<m>:INTegrat e:IOFFset? (see <a href="#">page 401</a> )	<input_offset> ::= DC offset correction in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:LINEar:G AIN <value> (see <a href="#">page 402</a> )	:FUNCTION<m>:LINEar:G AIN? (see <a href="#">page 402</a> )	<value> ::= 'A' in Ax + B, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:LINEar:O FFSet <value> (see <a href="#">page 403</a> )	:FUNCTION<m>:LINEar:O FFSet? (see <a href="#">page 403</a> )	<value> ::= 'B' in Ax + B, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:OFFSet <offset> (see <a href="#">page 404</a> )	:FUNCTION<m>:OFFSet? (see <a href="#">page 404</a> )	<offset> ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:OPERatio n <operation> (see <a href="#">page 405</a> )	:FUNCTION<m>:OPERatio n? (see <a href="#">page 407</a> )	<operation> ::= {ADD   SUBTract   MULTiply   DIVide   INTegrate   DIFF   FFT   SQRT   MAGNify   ABSolute   SQUare   LN   LOG   EXP   TEN   LOWPass   HIGHpass   AVERage   LINEar   MAXimum   MINimum   PEAK   MAXHold   MINHold   TRENd   BTIMing   BSTate} <m> ::= 1 to (# math functions) in NR1 format

**Table 99** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:RANGE <range> (see <a href="#">page 409</a> )	:FUNCTION<m>:RANGE? (see <a href="#">page 409</a> )	<p>&lt;range&gt; ::= the full-scale vertical axis value in NR3 format.</p> <p>The range for ADD, SUBT, MULT is 8E-6 to 800E+3. The range for the INTegrate function is 8E-9 to 400E+3.</p> <p>The range for the DIFF function is 80E-3 to 8.0E12 (depends on current sweep speed).</p> <p>The range for the FFT function is 8 to 800 dBV.</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:FUNCTION<m>:REFERenc e <level> (see <a href="#">page 410</a> )	:FUNCTION<m>:REFERenc e? (see <a href="#">page 410</a> )	<p>&lt;level&gt; ::= the value at center screen in NR3 format.</p> <p>The range of legal values is +/-10 times the current sensitivity of the selected function.</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:FUNCTION<m>:SCALe <scale value>[<suffix>] (see <a href="#">page 411</a> )	:FUNCTION<m>:SCALe? (see <a href="#">page 411</a> )	<p>&lt;scale value&gt; ::= integer in NR1 format</p> <p>&lt;suffix&gt; ::= {V   dB}</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:FUNCTION<m>:SMOoth:POINTs <points> (see <a href="#">page 412</a> )	:FUNCTION<m>:SMOoth:POINTs? (see <a href="#">page 412</a> )	<points> ::= odd integer in NR1 format
:FUNCTION<m>:SOURcel <source> (see <a href="#">page 413</a> )	:FUNCTION<m>:SOURcel? (see <a href="#">page 413</a> )	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;c&gt;   MATH&lt;c&gt;   BUS&lt;b&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;c&gt; ::= {1}, must be lower than &lt;m&gt;</p> <p>&lt;b&gt; ::= {1   2}</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>

**Table 99** :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:SOURCE2<source> (see page 415)	:FUNCTION<m>:SOURCE2? (see page 415)	<source> ::= {CHANnel<n>   NONE} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:TREND:MEA surement <type> (see page 416)	:FUNCTION<m>:TREND:MEA surement? (see page 416)	<type> ::= {VAVerage   ACRMs   VRATio   PERiod   FREQuency   PWIDth   NWIDTH   DUTYcycle   RISeTime   FALLtime} <m> ::= 1 to (# math functions) in NR1 format

**Introduction to :FUNCTION<m> Commands** The FUNCtion subsystem controls the math functions in the oscilloscope. Two math functions are available – the <m> in :FUNCTION<m> can be from 1 to 2. However, a dedicated FFT function is also available, see [Chapter 17, “FFT Commands,”](#) starting on page 359.

The math function operator, transform, filter, or visualization is selected using the :FUNCTION<m>:OPERation command. Depending on the selected operation, there may be other commands for specifying options for that operation. See [“:FUNCTION<m>:OPERation”](#) on page 405.

The SOURCE1, DISPLAY, RANGE, and OFFSET (or REFERENCE) commands apply to any function.

### Reporting the Setup

Use :FUNCTION<m>? to query setup information for the FUNCtion subsystem.

### Return Format

The following is a sample response from the :FUNCTION1? query. In this case, the query was issued following a \*RST command.

```
:FUNC1:OPER ADD;DISP 0;SOUR1 CHAN1;SOUR2 CHAN2;RANG +8.00E+00;OFFS +0.0E+00
```

**:FUNCTION<m>:AVERage:COUNT****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:AVERage:COUNT <count>`

`<count>` ::= an integer from 2 to 65536 in NR1 format

`<m>` ::= 1 to (# math functions) in NR1 format

The :FUNCTION<m>:AVERage:COUNT command sets the number of waveforms to be averaged together.

The number of averages can be set from 2 to 65536 in increments of powers of 2.

Increasing the number of averages will increase resolution and reduce noise.

**Query Syntax**    `:FUNCTION<m>:AVERage:COUNT?`

The :FUNCTION<m>:AVERage:COUNT? query returns the number of waveforms to be averaged together.

**Return Format**    `<count><NL>`

`<count>` ::= an integer from 2 to 65536 in NR1 format

**See Also**    · [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:BUS:CLOCK

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:BUS:CLOCK <source>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<source> ::= {DIGital<d>}`

`<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :FUNCTION<m>:BUS:CLOCK command selects the clock signal source for the Chart Logic Bus State operation.

**Query Syntax**    `:FUNCTION<m>:BUS:CLOCK?`

The :FUNCTION<m>:BUS:CLOCK query returns the source selected for the clock signal.

**Return Format**    `<source><NL>`

`<source> ::= {DIGital<d>}`

`<d> ::= 0 to (# digital channels - 1) in NR1 format`

**See Also**    • " [":FUNCTION<m>:OPERation](#)" on page 405

**:FUNCTION<m>:BUS:SLOPe****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:BUS:SLOPe <slope>`    `<m> ::= 1 to (# math functions) in NR1 format`    `<slope> ::= {NEGative | POSitive | EITHer}`

The :FUNCTION<m>:BUS:SLOPe command specifies the clock signal edge for the Chart Logic Bus State operation.

**Query Syntax**    `:FUNCTION<m>:BUS:SLOPe?`

The :FUNCTION<m>:BUS:SLOPe query returns the clock edge setting.

**Return Format**    `<slope><NL>`    `<slope> ::= {NEGative | POSitive | EITHer}`**See Also**    • [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:BUS:YINCrement

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:BUS:YINCrement <value>

<m> ::= 1 to (# math functions) in NR1 format

<value> ::= value per bus code, in NR3 format

The :FUNCTION<m>:BUS:YINCrement command specifies the value associated with each increment in Chart Logic Bus data.

**Query Syntax** :FUNCTION<m>:BUS:YINCrement?

The :FUNCTION<m>:BUS:YINCrement query returns the value associated with each increment in Chart Logic Bus data.

**Return Format** <value><NL>

<value> ::= value per bus code, in NR3 format

**See Also** • [":FUNCTION<m>:OPERation"](#) on page 405

**:FUNCTION<m>:BUS:YORigin****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:BUS:YORigin <value>`    `<m> ::= 1 to (# math functions) in NR1 format`    `<value> ::= value at bus code = 0, in NR3 format`

The :FUNCTION<m>:BUS:YORigin command specifies the value associated with Chart Logic Bus data equal to zero.

**Query Syntax**    `:FUNCTION<m>:BUS:YORigin?`

The :FUNCTION<m>:BUS:YORigin query returns the value for associated with data equal to zero.

**Return Format**    `<value><NL>`    `<value> ::= value at bus code = 0, in NR3 format`**See Also**    · [":FUNCTION<m>:OPERation"](#) on page 405

**:FUNCTION<m>:BUS:YUNits**

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:BUS:YUNits <units>

<m> ::= 1 to (# math functions) in NR1 format

<units> ::= {VOLT | AMPere | NONE}

The :FUNCTION<m>:BUS:YUNits command specifies the vertical units for the Chart Logic Bus operations.

**Query Syntax** :FUNCTION<m>:BUS:YUNits?

The :FUNCTION<m>:BUS:YUNits query returns the Chart Logic Bus vertical units.

**Return Format** <units><NL>

<units> ::= {VOLT | AMP | NONE}

**See Also** • [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:CLEar

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:CLEar`

When the :FUNCTION<m>:OPERation is AVERage, MAXHold, or MINHold, the :FUNCTION<m>:CLEar command clears the number of evaluated waveforms.

**See Also**    • [":FUNCTION<m>:AVERage:COUNT"](#) on page 384

## :FUNCTION<m>:DISPlay

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:DISPlay <display>

```
<m> ::= 1 to (# math functions) in NR1 format
<display> ::= {{1 | ON} | {0 | OFF}}
```

The :FUNCTION<m>:DISPlay command turns the display of the function on or off. When ON is selected, the function performs as specified using the other FUNCTION commands. When OFF is selected, function is neither calculated nor displayed.

**Query Syntax** :FUNCTION<m>:DISPlay?

The :FUNCTION<m>:DISPlay? query returns whether the function display is on or off.

**Return Format** <display><NL>

```
<display> ::= {1 | 0}
```

**See Also**

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":VIEW"](#) on page 242
- "[":BLANK"](#) on page 214
- "[":STATus"](#) on page 239

## :FUNCTION<m>[:FFT]:CENTer



(see [page 1334](#))

**Command Syntax**

```
:FUNCTION<m> [:FFT] :CENTer <frequency>
<m> ::= 1 to (# math functions) in NR1 format
<frequency> ::= the current center frequency in NR3 format. The range
of legal values is from -25 GHz to 25 GHz.
```

The :FUNCTION<m>[:FFT]:CENTer command sets the center frequency when FFT (Fast Fourier Transform) is selected.

**Query Syntax**

```
:FUNCTION<m> [:FFT] :CENTer?
```

The :FUNCTION<m>[:FFT]:CENTer? query returns the current center frequency in Hertz.

**Return Format**

```
<frequency><NL>
<frequency> ::= the current center frequency in NR3 format. The range
of legal values is from -25 GHz to 25 GHz.
```

### NOTE

After a \*RST (Reset) or :AUToscale command, the values returned by the :FUNCTION<m>[:FFT]:CENTer? and :FUNCTION<m>:SPAN? queries depend on the current :TIMEbase:RANGE value. Once you change either the :FUNCTION<m>[:FFT]:CENTer or :FUNCTION<m>:SPAN value, they no longer track the :TIMEbase:RANGE value.

### See Also

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":FUNCTION<m>\[:FFT\]:SPAN"](#) on page 396
- "[":TIMEbase:RANGE"](#) on page 1039
- "[":TIMEbase:SCALe"](#) on page 1041

## :FUNCTION<m>[:FFT]:FREQuency:STARt

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m> [:FFT] :FREQuency:STARt <frequency>

<m> ::= 1 to (# math functions) in NR1 format

<frequency> ::= the start frequency in NR3 format.

The :FUNCTION<m>[:FFT]:FREQuency:STARt command sets the start frequency in the FFT (Fast Fourier Transform) math function's displayed range.

The FFT (Fast Fourier Transform) math function's displayed range can also be set with the :FUNCTION<m>[:FFT]:CENTer and :FUNCTION<m>[:FFT]:SPAN commands.

**Query Syntax** :FUNCTION<m> [:FFT] :FREQuency:STARt?

The :FUNCTION<m>[:FFT]:FREQuency:STARt? query returns the current start frequency in Hertz.

**Return Format** <frequency><NL>

<frequency> ::= the start frequency in NR3 format.

**See Also**

- "[:FUNCTION<m>\[:FFT\]:FREQuency:STOP](#)" on page 394
- "[:FUNCTION<m>\[:FFT\]:CENTer](#)" on page 392
- "[:FUNCTION<m>\[:FFT\]:SPAN](#)" on page 396

## :FUNCTION<m>[:FFT]:FREQuency:STOP

**N** (see [page 1334](#))

**Command Syntax**

```
:FUNCTION<m> [:FFT] :FREQuency:STOP <frequency>
<m> ::= 1 to (# math functions) in NR1 format
<frequency> ::= the stop frequency in NR3 format.
```

The :FUNCTION<m>[:FFT]:FREQuency:STOP command sets the stop frequency in the FFT (Fast Fourier Transform) math function's displayed range.

The FFT (Fast Fourier Transform) math function's displayed range can also be set with the :FUNCTION<m>[:FFT]:CENTer and :FUNCTION<m>[:FFT]:SPAN commands.

**Query Syntax**

```
:FUNCTION<m> [:FFT] :FREQuency:STOP?
```

The :FUNCTION<m>[:FFT]:FREQuency:STOP? query returns the current stop frequency in Hertz.

**Return Format**

```
<frequency><NL>
<frequency> ::= the stop frequency in NR3 format.
```

**See Also**

- "[:FUNCTION<m>\[:FFT\]:FREQuency:STARt](#)" on page 393
- "[:FUNCTION<m>\[:FFT\]:CENTer](#)" on page 392
- "[:FUNCTION<m>\[:FFT\]:SPAN](#)" on page 396

## :FUNCTION<m>[:FFT]:GATE



(see [page 1334](#))

**Command Syntax**    `:FUNCTION<m> [:FFT] :GATE <gating>`

`<m> ::= 1-4 in NRI format`

`<gating> ::= {NONE | ZOOM}`

The :FUNCTION<m>[:FFT]:GATE command specifies whether the FFT is performed on the Main time base window (NONE) or the ZOOM window when the zoomed time base is displayed.

**Query Syntax**    `:FUNCTION<m> [:FFT] :GATE?`

The :FUNCTION<m>[:FFT]:GATE? query returns the gate setting.

**Return Format**    `<gating><NL>`

`<gating> ::= {NONE | ZOOM}`

**See Also**

- [":FUNCTION<m>\[:FFT\]:VTPe"](#) on page 397
- [":FUNCTION<m>\[:FFT\]:WINDow"](#) on page 398
- ["Introduction to :FUNCTION<m> Commands"](#) on page 383
- [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>[:FFT]:SPAN

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:FUNCTION&lt;m&gt; [:FFT] :SPAN &lt;span&gt;</code>
	<code>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</code>
	<code>&lt;span&gt; ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.</code>
	If you set the frequency span to a value outside of the legal range, the step size is automatically set to the nearest legal value.

The :FUNCTION<m>[:FFT]:SPAN command sets the frequency span of the display (left graticule to right graticule) when FFT (Fast Fourier Transform) is selected.

<b>Query Syntax</b>	<code>:FUNCTION&lt;m&gt; [:FFT] :SPAN?</code>
	The :FUNCTION<m>[:FFT]:SPAN? query returns the current frequency span in Hertz.

**NOTE** After a \*RST (Reset) or :AUToscale command, the values returned by the :FUNCTION<m>[:FFT]:CENTer? and :FUNCTION<m>:SPAN? queries depend on the current :TIMEbase:RANGE value. Once you change either the :FUNCTION<m>[:FFT]:CENTer or :FUNCTION<m>:SPAN value, they no longer track the :TIMEbase:RANGE value.

<b>Return Format</b>	<code>&lt;span&gt;&lt;NL&gt;</code>
	<code>&lt;span&gt; ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :FUNCTION&lt;m&gt; Commands</a>" on page 383</li> <li>· "<a href="#">:FUNCTION&lt;m&gt;[:FFT]:CENTer</a>" on page 392</li> <li>· "<a href="#">:TIMEbase:RANGE</a>" on page 1039</li> <li>· "<a href="#">:TIMEbase:SCALe</a>" on page 1041</li> </ul>

**:FUNCTION<m>[:FFT]:VTYPe**

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m> [:FFT] :VTYPe <units>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<units> ::= {DECibel | VRMS}`

The `:FUNCTION<m>[:FFT]:VTYPe` command specifies FFT vertical units as DECibel or VRMS.

**Query Syntax**    `:FUNCTION<m> [:FFT] :VTYPe?`

The `:FUNCTION<m>[:FFT]:VTYPe?` query returns the current FFT vertical units.

**Return Format**    `<units><NL>`

`<units> ::= {DEC | VRMS}`

- See Also**
- "[:FUNCTION<m>\[:FFT\]:GATE](#)" on page 395
  - "[Introduction to :FUNCTION<m> Commands](#)" on page 383
  - "[:FUNCTION<m>:OPERation](#)" on page 405

## :FUNCTION<m>[:FFT]:WINDOW

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m> [:FFT] :WINDOW <window>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<window> ::= {RECTangular | HANNing | FLATtop | BHARris}`

The :FUNCTION<m>[:FFT]:WINDOW command allows the selection of four different windowing transforms or operations for the FFT (Fast Fourier Transform) function.

The FFT operation assumes that the time record repeats. Unless an integral number of sampled waveform cycles exist in the record, a discontinuity is created between the end of one record and the beginning of the next. This discontinuity introduces additional frequency components about the peaks into the spectrum. This is referred to as leakage. To minimize leakage, windows that approach zero smoothly at the start and end of the record are employed as filters to the FFTs. Each window is useful for certain classes of input signals.

- RECTangular – useful for transient signals, and signals where there are an integral number of cycles in the time record.
- HANNing – useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together, or for making frequency measurements. This is the default window.
- FLATtop – best for making accurate amplitude measurements of frequency peaks.
- BHARris (Blackman-Harris) – reduces time resolution compared to the rectangular window, but it improves the capacity to detect smaller impulses due to lower secondary lobes (provides minimal spectral leakage).

**Query Syntax**    `:FUNCTION<m> [:FFT] :WINDOW?`

The :FUNCTION<m>[:FFT]:WINDOW? query returns the value of the window selected for the FFT function.

**Return Format**    `<window><NL>`

`<window> ::= {RECT | HANN | FLAT | BHAR}`

**See Also**

- "[:FUNCTION<m>\[:FFT\]:GATE](#)" on page 395
- "["Introduction to :FUNCTION<m> Commands"](#) on page 383

## :FUNCTION<m>:FREQuency:HIGHpass

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:FREQuency:HIGHpass <3dB\_freq>

<m> ::= 1 to (# math functions) in NR1 format

<3dB\_freq> ::= -3dB cutoff frequency value in NR3 format

The :FUNCTION<m>:FREQuency:HIGHpass command sets the high-pass filter's -3 dB cutoff frequency.

The high-pass filter is a single-pole high pass filter.

**Query Syntax** :FUNCTION<m>:FREQuency:HIGHpass?

The :FUNCTION<m>:FREQuency:HIGHpass query returns the high-pass filter's cutoff frequency.

**Return Format** <3dB\_freq><NL>

<3dB\_freq> ::= -3dB cutoff frequency value in NR3 format

**See Also** • [":FUNCTION<m>:OPERation"](#) on page 405

**:FUNCTION<m>:FREQuency:LOWPass****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:FREQuency:LOWPass <3dB_freq>`    `<m> ::= 1 to (# math functions) in NR1 format`    `<3dB_freq> ::= -3dB cutoff frequency value in NR3 format`

The :FUNCTION<m>:FREQuency:LOWPass command sets the low-pass filter's -3 dB cutoff frequency.

The low-pass filter is a 4th order Bessel-Thompson filter.

**Query Syntax**    `:FUNCTION<m>:FREQuency:LOWPass?`

The :FUNCTION<m>:FREQuency:LOWPass query returns the low-pass filter's cutoff frequency.

**Return Format**    `<3dB_freq><NL>`    `<3dB_freq> ::= -3dB cutoff frequency value in NR3 format`**See Also**

- [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:INTegrate:IOFFset

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:INTegrate:IOFFset <input_offset>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<input_offset> ::= DC offset correction in NR3 format.`

The :FUNCTION<m>:INTegrate:IOFFset command lets you enter a DC offset correction factor for the integrate math waveform input signal. This DC offset correction lets you level a "ramp"ed waveform.

**Query Syntax**    `:FUNCTION<m>:INTegrate:IOFFset?`

The :FUNCTION<m>:INTegrate:IOFFset? query returns the current input offset value.

**Return Format**    `<input_offset><NL>`

`<input_offset> ::= DC offset correction in NR3 format.`

**See Also**

- ["Introduction to :FUNCTION<m> Commands"](#) on page 383
- [":FUNCTION<m>:OPERation"](#) on page 405

**:FUNCTION<m>:LINEar:GAIN****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:LINEar:GAIN <value>`    `<m> ::= 1 to (# math functions) in NR1 format`    `<value> ::= 'A' in Ax + B, value in NR3 format`

The :FUNCTION<m>:LINEar:GAIN command specifies the 'A' value in the Ax + B operation.

**Query Syntax**    `:FUNCTION<m>:LINEar:GAIN?`

The :FUNCTION<m>:LINEar:GAIN query returns the gain value.

**Return Format**    `<value><NL>`    `<value> ::= 'A' in Ax + B, value in NR3 format`**See Also**

- [":FUNCTION<m>:OPERation"](#) on page 405

**:FUNCTION<m>:LINear:OFFSet**

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:LINear:OFFSet <value>

<m> ::= 1 to (# math functions) in NR1 format

<value> ::= 'B' in Ax + B, value in NR3 format

The :FUNCTION<m>:LINear:OFFSet command specifies the 'B' value in the Ax + B operation.

**Query Syntax** :FUNCTION<m>:LINear:OFFSet?

The :FUNCTION<m>:LINear:OFFSet query returns the offset value.

**Return Format** <value><NL>

<value> ::= 'B' in Ax + B, value in NR3 format

**See Also** • [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:OFFSet

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:OFFSet <offset>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<offset> ::= the value at center screen in NR3 format.`

The :FUNCTION<m>:OFFSet command sets the voltage or vertical value represented at center screen for the selected function. The range of legal values is generally +/-10 times the current scale of the selected function, but will vary by function. If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value.

### NOTE

The :FUNCTION<m>:OFFSet command is equivalent to the :FUNCTION<m>:REFERENCE command.

**Query Syntax**    `:FUNCTION<m>:OFFSet?`

The :FUNCTION<m>:OFFSet? query outputs the current offset value for the selected function.

**Return Format**    `<offset><NL>`

`<offset> ::= the value at center screen in NR3 format.`

**See Also**

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":FUNCTION<m>:RANGE](#)" on page 409
- "[":FUNCTION<m>:REFERENCE](#)" on page 410
- "[":FUNCTION<m>:SCALE](#)" on page 411

## :FUNCTION<m>:OPERation

**N** (see [page 1334](#))

**Command Syntax** :FUNCTION<m>:OPERation <operation>

```
<m> ::= 1 to (# math functions) in NR1 format
<operation> ::= {ADD | SUBTract | MULTIply | DIVide | DIFF | INTegrate
| FFT | SQRT | MAGNify | ABSolute | SQUare | LN | LOG | EXP | TEN
| LOWPass | HIGHpass | AVERage | SMOOTH | ENvelope | LINear
| MAXimum | MINimum | PEAK | MAXHold | MINHold | TRENd | BTIMing
| BSTate}
```

The :FUNCTION<m>:OPERation command sets the desired waveform math operator, transform, filter or visualization:

- Operators:
  - ADD – Source1 + source2.
  - SUBTract – Source1 - source2.
  - MULTIply – Source1 \* source2.
  - DIVide – Source1 / source2.
- Operators perform their function on two analog channel sources.
- Transforms:
  - DIFF – Differentiate
  - INTegrate – The INTegrate:IOFFset command lets you specify a DC offset correction factor.
  - FFT – The SPAN, CENTer, VTYPE, and WINDow commands are used for FFT functions. When FFT is selected, the horizontal cursors change from time to frequency (Hz), and the vertical cursors change from volts to decibel (dB).
  - LINear – Ax + B – The LINear commands set the gain (A) and offset (B) values for this function.
  - SQUare
  - SQRT – Square root
  - ABSolute – Absolute Value
  - LOG – Common Logarithm
  - LN – Natural Logarithm
  - EXP – Exponential ( $e^x$ )
  - TEN – Base 10 exponential ( $10^x$ )
- Transforms operate on a single analog channel source or on lower math functions.
- Filters:

- LOWPass – Low pass filter – The FREQuency:LOWPass command sets the -3 dB cutoff frequency.
- HIGHpass – High pass filter – The FREQuency:HIGHpass command sets the -3 dB cutoff frequency.
- AVERage – Averaged value – The AVERage:COUNt command specifies the number of averages.

Unlike acquisition averaging, the math averaging operator can be used to average the data on a single analog input channel or math function.

If acquisition averaging is also used, the analog input channel data is averaged and the math function averages it again. You can use both types of averaging to get a certain number of averages on all waveforms and an increased number of averages on a particular waveform.

Averages are calculated using a "decaying average" approximation, where:

$$\text{next\_average} = \text{current\_average} + (\text{new\_data} - \text{current\_average})/N$$

Where N starts at 1 for the first acquisition and increments for each following acquisition until it reaches the selected number of averages, where it holds.

- SMOoth – Smoothing – The resulting math waveform is the selected source with a normalized rectangular (boxcar) FIR filter applied.

The boxcar filter is a moving average of adjacent waveform points, where the number of adjacent points is specified by the SMOoth:POINts command. You can choose an odd number of points, from 3 to 999.

The smoothing operator limits the bandwidth of the source waveform. The smoothing operator can be used, for example, to smooth measurement trend waveforms.

- ENVelope – Envelope – The resulting math waveform is the amplitude envelope for an amplitude modulated (AM) input signal.

This function uses a Hilbert transform to get the real (in-phase, I) and imaginary (quadrature, Q) parts of the input signal and then performs a square root of the sum of the real and imaginary parts to get the demodulated amplitude envelope waveform.

Filters operate on a single analog channel source or on a lower math function.

- Visualizations:
  - MAGNify – Operates on a single analog channel source or on a lower math function.
  - MAXimum – This operator is like the MAXHold operator without the hold. The maximum vertical values found at each horizontal bucket are used to build a waveform.

- MINimum – This operator is like the MINHold operator without the hold. The minimum vertical values found at each horizontal bucket are used to build a waveform.
- PEAK – The PEAK operator is like the MAXimum operator minus the MINimum operator. At each horizontal bucket, the minimum vertical values found are subtracted from the maximum vertical values found to build a waveform.
- MAXHold – Operates on a single analog channel source or on a lower math function. The Max Hold (or Max Envelope) operator records the maximum vertical values found at each horizontal bucket across multiple analysis cycles and uses those values to build a waveform.
- MINHold – Operates on a single analog channel source or on a lower math function. The Min Hold (or Min Envelope) operator records the minimum vertical values found at each horizontal bucket across multiple analysis cycles and uses those values to build a waveform.
- TRENd – Measurement trend – Operates on a single analog channel source. The TRENd:MEASurement command selects the measurement whose trend you want to measure.
- BTIMing – Chart logic bus timing – Operates on a bus made up of digital channels. The BUS:YINcrement, BUS:YORigin, and BUS:YUNits commands specify function values.
- BSTate – Chart logic bus state – Operates on a bus made up of digital channels. The BUS:YINcrement, BUS:YORigin, and BUS:YUnit commands specify function values. The BUS:CLOCK and BUS:SLOPe commands specify the clock source and edge.

**Query Syntax** :FUNCTION<m>:OPERation?

The :FUNCTION<m>:OPERation? query returns the current operation for the selected function.

**Return Format** <operation><NL>

```
<operation> ::= {ADD | SUBT | MULT | DIV | INT | DIFF | FFT | SQRT
| MAGN | ABS | SQU | LN | LOG | EXP | TEN | LOWP | HIGH | AVER
| SMO | ENV | LIN | MAX | MIN | PEAK | | MAXH | MINH | TREN
| BTIM | BST}
```

- See Also**
- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
  - "[":FUNCTION<m>:SOURce1](#)" on page 413
  - "[":FUNCTION<m>:SOURce2](#)" on page 415
  - "[":FUNCTION<m>:INTegrate:OFFset](#)" on page 401
  - "[":FUNCTION<m>\[:FFT\]:SPAN](#)" on page 396
  - "[":FUNCTION<m>\[:FFT\]:CENTer](#)" on page 392
  - "[":FUNCTION<m>\[:FFT\]:VTYPE](#)" on page 397

- "[:FUNCTION<m>\[:FFT\]:WINDOW](#)" on page 398
- "[:FUNCTION<m>:LINEar:GAIN](#)" on page 402
- "[:FUNCTION<m>:LINEar:OFFSet](#)" on page 403
- "[:FUNCTION<m>:FREQuency:LOWPass](#)" on page 400
- "[:FUNCTION<m>:FREQuency:HIGHpass](#)" on page 399
- "[:FUNCTION<m>:AVERage:COUNT](#)" on page 384
- "[:FUNCTION<m>:TRENd:MEASurement](#)" on page 416
- "[:FUNCTION<m>:BUS:YINCrement](#)" on page 387
- "[:FUNCTION<m>:BUS:YORigin](#)" on page 388
- "[:FUNCTION<m>:BUS:YUNits](#)" on page 389
- "[:FUNCTION<m>:BUS:CLOCK](#)" on page 385
- "[:FUNCTION<m>:BUS:SLOPe](#)" on page 386

**:FUNCTION<m>:RANGE****N** (see [page 1334](#))**Command Syntax**    `:FUNCTION<m>:RANGE <range>`    `<m> ::= 1 to (# math functions) in NR1 format`    `<range> ::= the full-scale vertical axis value in NR3 format.`

The :FUNCTION<m>:RANGE command defines the full-scale vertical axis for the selected function.

**Query Syntax**    `:FUNCTION<m>:RANGE?`

The :FUNCTION<m>:RANGE? query returns the current full-scale range value for the selected function.

**Return Format**    `<range><NL>`    `<range> ::= the full-scale vertical axis value in NR3 format.`**See Also**

- "Introduction to :FUNCTION<m> Commands" on page 383
- "[:FUNCTION<m>:SCALe](#)" on page 411

## :FUNCTION<m>:REFERENCE

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:REFERENCE <level>`

`<m> ::= 1 to (# math functions) in NR1 format`

`<level> ::= the current reference level in NR3 format.`

The :FUNCTION<m>:REFERENCE command sets the voltage or vertical value represented at center screen for the selected function. The range of legal values is generally +/-10 times the current scale of the selected function, but will vary by function. If you set the reference level to a value outside of the legal range, the level is automatically set to the nearest legal value.

### NOTE

The FUNCTION:REFERENCE command is equivalent to the :FUNCTION<m>:OFFSET command.

**Query Syntax**    `:FUNCTION<m>:REFERENCE?`

The :FUNCTION<m>:REFERENCE? query outputs the current reference level value for the selected function.

**Return Format**    `<level><NL>`

`<level> ::= the current reference level in NR3 format.`

**See Also**

- "Introduction to :FUNCTION<m> Commands" on page 383
- "[:FUNCTION<m>:OFFSET](#)" on page 404
- "[:FUNCTION<m>:RANGE](#)" on page 409
- "[:FUNCTION<m>:SCALE](#)" on page 411

## :FUNCTION<m>:SCALE

**N** (see [page 1334](#))

**Command Syntax**

```
:FUNCTION<m>:SCALE <scale value>[<suffix>]
<m> ::= 1 to (# math functions) in NR1 format
<scale value> ::= integer in NR1 format
<suffix> ::= {v | dB}
```

The :FUNCTION<m>:SCALE command sets the vertical scale, or units per division, of the selected function. Legal values for the scale depend on the selected function.

**Query Syntax**

```
:FUNCTION<m>:SCALE?
```

The :FUNCTION<m>:SCALE? query returns the current scale value for the selected function.

**Return Format**

```
<scale value><NL>
<scale value> ::= integer in NR1 format
```

**See Also**

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":FUNCTION<m>:RANGE](#)" on page 409

**:FUNCTION<m>:SMOoth:POINTS**

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:SMOoth:POINTS <points>`  
`<points> ::= odd integer in NR1 format`

When the :FUNCTION<m>:OPERation is SMOoth, the :FUNCTION<m>:SMOoth:POINTS command sets the number of smoothing points to use.

You can choose an odd number of points, from 3 up to half of the measurement record or precision analysis record.

**Query Syntax**    `:FUNCTION<m>:SMOoth:POINTS?`

The :FUNCTION<m>:SMOoth:POINTS? query returns the number of smoothing points specified.

**Return Format**    `<points><NL>`

**See Also**    • [":FUNCTION<m>:OPERation"](#) on page 405

## :FUNCTION<m>:SOURce1

**N** (see [page 1334](#))

### Command Syntax

```
:FUNCTION<m>:SOURce1 <value>
<m> ::= 1 to (# math functions) in NR1 format
<value> ::= {CHANnel<n> | FUNCTION<c> | MATH<c> | BUS<b>}
<n> ::= 1 to (# analog channels) in NR1 format
<c> ::= {1}, must be lower than <m>
<b> ::= {1 | 2}
```

The :FUNCTION<m>:SOURce1 command is used for any :FUNCTION<m>:OPERation selection. This command selects the first source for the operator math functions or the single source for the transform functions, filter functions, or visualization functions.

The FUNCTION<c> or MATH<c> parameters are available for the transform functions, filter functions, and the magnify visualization function (see "[Introduction to :FUNCTION<m> Commands](#)" on page 383) when <c> is lower than <m>.

In other words, higher math functions can operate on lower math functions when using operators other than the simple arithmetic operations (+, -, \*, /). For example, if :FUNCTION1:OPERation is a SUBTract of CHANnel1 and CHANnel2, the :FUNCTION2:OPERation could be set up as a FFT operation on the FUNCTION1 source. These are called cascaded math functions.

To cascade math functions, select the lower math function using the :FUNCTION<m>:SOURce1 command.

When cascading math functions, to get the most accurate results, be sure to vertically scale lower math functions so that their waveforms take up the full screen without being clipped.

The BUS<m> parameter is available for the bus charting visualization functions.

### NOTE

Another shorthand notation for SOURce1 in this command/query (besides SOUR1) is SOUR.

### Query Syntax

```
:FUNCTION<m>:SOURce1?
```

The :FUNCTION<m>:SOURce1? query returns the current source1 for function operations.

### Return Format

```
<value><NL>
<value> ::= {CHAN<n> | FUNCTION<c> | MATH<c> | BUS<b>}
<n> ::= 1 to (# analog channels) in NR1 format
```

```
<c> ::= {1}, must be lower than <m>
<b> ::= {1 | 2}
```

- See Also**
- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
  - "[":FUNCTION<m>:OPERation](#)" on page 405

## :FUNCTION<m>:SOURce2

**N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:SOURce2 <value>`

```
<m> ::= 1 to (# math functions) in NR1 format
<value> ::= {CHANnel<n> | NONE}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :FUNCTION<m>:SOURce2 command specifies the second source for math operator functions that have two sources. (The :FUNCTION<m>:SOURce1 command specifies the first source.)

The :FUNCTION<m>:SOURce2 setting is not used for the transform functions, filter functions, or visualization functions (except when the measurement trend visualization's measurement requires two sources).

**Query Syntax**    `:FUNCTION<m>:SOURce2?`

The :FUNCTION<m>:SOURce2? query returns the currently specified second source for math operations.

**Return Format**    `<value><NL>`

```
<value> ::= {CHAN<n> | NONE}
<n> ::= 1 to (# analog channels) in NR1 format
```

**See Also**

- "Introduction to :FUNCTION<m> Commands" on page 383

- "[:FUNCTION<m>:OPERation](#)" on page 405

- "[:FUNCTION<m>:SOURce1](#)" on page 413

**:FUNCTION<m>:TRENd:MEASurement****N** (see [page 1334](#))

**Command Syntax**    `:FUNCTION<m>:TRENd:MEASurement <type>`  
`<m> ::= 1 to (# math functions) in NR1 format`  
`<type> ::= {VAVerage | ACRMs | VRATio | PERiod | FREQuency | PWIDth`  
`| NWIDth | DUTYcycle | RISetime | FALLtime}`

The :FUNCTION<m>:TRENd:MEASurement command selects the measurement whose trend is shown in the math waveform.

**Query Syntax**    `:FUNCTION<m>:TRENd:MEASurement?`

The :FUNCTION<m>:TRENd:MEASurement query returns the selected measurement.

**Return Format**    `<type><NL>`  
`<type> ::= {VAV | ACRM | VRAT | PER | FREQ | PWID | NWID | DUTY`  
`| RIS | FALL}`

**See Also**    • [":FUNCTION<m>:OPERation"](#) on page 405

# 19 :HARDcopy Commands

Set and query the selection of hardcopy device and formatting options. See "[Introduction to :HARDcopy Commands](#)" on page 418.

**Table 100**:HARDcopy Commands Summary

Command	Query	Options and Query Returns
:HARDcopy:AREA <area> (see <a href="#">page 419</a> )	:HARDcopy:AREA? (see <a href="#">page 419</a> )	<area> ::= SCReen
:HARDcopy:APRinter <active_printer> (see <a href="#">page 420</a> )	:HARDcopy:APRinter? (see <a href="#">page 420</a> )	<active_printer> ::= {<index>   <name>} <index> ::= integer index of printer in list <name> ::= name of printer in list
:HARDcopy:FACTors {{0   OFF}   {1   ON}} (see <a href="#">page 421</a> )	:HARDcopy:FACTors? (see <a href="#">page 421</a> )	{0   1}
:HARDcopy:FFEed {{0   OFF}   {1   ON}} (see <a href="#">page 422</a> )	:HARDcopy:FFEed? (see <a href="#">page 422</a> )	{0   1}
:HARDcopy:INKSaver { {0   OFF}   {1   ON}} (see <a href="#">page 423</a> )	:HARDcopy:INKSaver? (see <a href="#">page 423</a> )	{0   1}
:HARDcopy:LAYout <layout> (see <a href="#">page 424</a> )	:HARDcopy:LAYout? (see <a href="#">page 424</a> )	<layout> ::= {LANDscape   PORTRait}
:HARDcopy:NETWork:ADD Ress <address> (see <a href="#">page 425</a> )	:HARDcopy:NETWork:ADD Ress? (see <a href="#">page 425</a> )	<address> ::= quoted ASCII string
:HARDcopy:NETWork:APP Ly (see <a href="#">page 426</a> )	n/a	n/a
:HARDcopy:NETWork:DOM ain <domain> (see <a href="#">page 427</a> )	:HARDcopy:NETWork:DOM ain? (see <a href="#">page 427</a> )	<domain> ::= quoted ASCII string

**Table 100**:HARDcopy Commands Summary (continued)

Command	Query	Options and Query Returns
:HARDcopy:NETWork:PAS Sword <password> (see <a href="#">page 428</a> )	n/a	<password> ::= quoted ASCII string
:HARDcopy:NETWork:SLO T <slot> (see <a href="#">page 429</a> )	:HARDcopy:NETWork:SLO T? (see <a href="#">page 429</a> )	<slot> ::= {NET0   NET1}
:HARDcopy:NETWork:USE Rname <username> (see <a href="#">page 430</a> )	:HARDcopy:NETWork:USE Rname? (see <a href="#">page 430</a> )	<username> ::= quoted ASCII string
:HARDcopy:PAlette <palette> (see <a href="#">page 431</a> )	:HARDcopy:PAlette? (see <a href="#">page 431</a> )	<palette> ::= {COLOR   GRAYscale   NONE}
n/a	:HARDcopy:PRINTER:LIS T? (see <a href="#">page 432</a> )	<list> ::= [<printer_spec>] ... [<printer_spec>] <printer_spec> ::= "<index>,<active>,<name>;" <index> ::= integer index of printer <active> ::= {Y   N} <name> ::= name of printer
:HARDcopy:START (see <a href="#">page 433</a> )	n/a	n/a

**Introduction to :HARDcopy Commands** The HARDcopy subsystem provides commands to set and query the selection of hardcopy device and formatting options such as inclusion of instrument settings (FACTors) and generation of formfeed (FFEed).

:HARDC is an acceptable short form for :HARDcopy.

#### Reporting the Setup

Use :HARDcopy? to query setup information for the HARDcopy subsystem.

#### Return Format

The following is a sample response from the :HARDcopy? query. In this case, the query was issued following the \*RST command.

```
:HARD:APR "";AREA SCR;FACT 0;FFE 0;INKS 1;PAL NONE;LAY PORT
```

## :HARDcopy:AREA

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:AREA <area>`  
                      `<area> ::= SCReen`

The :HARDcopy:AREA command controls what part of the display area is printed. Currently, the only legal choice is SCReen.

**Query Syntax**    `:HARDcopy:AREA?`

The :HARDcopy:AREA? query returns the selected display area.

**Return Format**    `<area><NL>`  
                      `<area> ::= SCR`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[":HARDcopy:STARt](#)" on page 433
- "[":HARDcopy:APRinter](#)" on page 420
- "[":HARDcopy:PRINTER:LIST](#)" on page 432
- "[":HARDcopy:FACTors](#)" on page 421
- "[":HARDcopy:FFEed](#)" on page 422
- "[":HARDcopy:INKSaver](#)" on page 423
- "[":HARDcopy:LAYout](#)" on page 424
- "[":HARDcopy:PALETTE](#)" on page 431

## :HARDcopy:APRinter

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:APRinter <active_printer>`

`<active_printer> ::= {<index> | <name>}`

`<index> ::= integer index of printer in list`

`<name> ::= name of printer in list`

The :HARDcopy:APRinter command sets the active printer.

**Query Syntax**    `:HARDcopy:APRinter?`

The :HARDcopy:APRinter? query returns the name of the active printer.

**Return Format**    `<name><NL>`

`<name> ::= name of printer in list`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418

- "[:HARDcopy:PRINTER:LIST](#)" on page 432

- "[:HARDcopy:STARt](#)" on page 433

## :HARDcopy:FACTors

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:FACTors <factors>`  
`<factors> ::= {{OFF | 0} | {ON | 1}}`

The HARDcopy:FACTors command controls whether the scale factors are output on the hardcopy dump.

**Query Syntax**    `:HARDcopy:FACTors?`

The :HARDcopy:FACTors? query returns a flag indicating whether oscilloscope instrument settings are output on the hardcopy.

**Return Format**    `<factors><NL>`  
`<factors> ::= {0 | 1}`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[:HARDcopy:STARt](#)" on page 433
- "[:HARDcopy:FFEed](#)" on page 422
- "[:HARDcopy:INKSaver](#)" on page 423
- "[:HARDcopy:LAYout](#)" on page 424
- "[:HARDcopy:PAlette](#)" on page 431

**:HARDcopy:FFEed****N** (see [page 1334](#))**Command Syntax**    `:HARDcopy:FFEed <ffeed>``<ffeed> ::= {{OFF | 0} | {ON | 1}}`

The HARDcopy:FFEed command controls whether a formfeed is output between the screen image and factors of a hardcopy dump.

**Query Syntax**    `:HARDcopy:FFEed?`

The :HARDcopy:FFEed? query returns a flag indicating whether a formfeed is output at the end of the hardcopy dump.

**Return Format**    `<ffeed><NL>``<ffeed> ::= {0 | 1}`

- See Also**
- "[Introduction to :HARDcopy Commands](#)" on page 418
  - "[:HARDcopy:STARt](#)" on page 433
  - "[:HARDcopy:FACTors](#)" on page 421
  - "[:HARDcopy:INKSaver](#)" on page 423
  - "[:HARDcopy:LAYout](#)" on page 424
  - "[:HARDcopy:PAlette](#)" on page 431

**:HARDcopy:INKSaver****N** (see [page 1334](#))**Command Syntax**    `:HARDcopy:INKSaver <value>``<value> ::= {{OFF | 0} | {ON | 1}}`

The HARDcopy:INKSaver command controls whether the graticule colors are inverted or not.

**Query Syntax**    `:HARDcopy:INKSaver?`

The :HARDcopy:INKSaver? query returns a flag indicating whether graticule colors are inverted or not.

**Return Format**    `<value><NL>``<value> ::= {0 | 1}`

- See Also**
- "[Introduction to :HARDcopy Commands](#)" on page 418
  - "[:HARDcopy:STARt](#)" on page 433
  - "[:HARDcopy:FACTors](#)" on page 421
  - "[:HARDcopy:FFEed](#)" on page 422
  - "[:HARDcopy:LAYout](#)" on page 424
  - "[:HARDcopy:PAlette](#)" on page 431

## :HARDcopy:LAYout

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:LAYout <layout>`

`<layout> ::= {LANDscape | PORTrait}`

The :HARDcopy:LAYout command sets the hardcopy layout mode.

**Query Syntax**    `:HARDcopy:LAYout?`

The :HARDcopy:LAYout? query returns the selected hardcopy layout mode.

**Return Format**    `<layout><NL>`

`<layout> ::= {LAND | PORT}`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418

- "[:HARDcopy:STARt](#)" on page 433

- "[:HARDcopy:FACTors](#)" on page 421

- "[:HARDcopy:PALETTE](#)" on page 431

- "[:HARDcopy:FFEed](#)" on page 422

- "[:HARDcopy:INKSaver](#)" on page 423

## :HARDcopy:NETWork:ADDRess

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:NETWork:ADDRess <address>`  
`<address> ::= quoted ASCII string`

The :HARDcopy:NETWork:ADDRess command sets the address for a network printer slot. The address is the server/computer name and the printer's share name in the \\server\share format.

The network printer slot is selected by the :HARDcopy:NETWork:SLOT command.

To apply the entered address, use the :HARDcopy:NETWork:APPLy command.

**Query Syntax**    `:HARDcopy:NETWork:ADDRess?`

The :HARDcopy:NETWork:ADDRess? query returns the specified address for the currently selected network printer slot.

**Return Format**    `<address><NL>`  
`<address> ::= quoted ASCII string`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[":HARDcopy:NETWork:SLOT](#)" on page 429
- "[":HARDcopy:NETWork:APPLy](#)" on page 426
- "[":HARDcopy:NETWork:DOMAIN](#)" on page 427
- "[":HARDcopy:NETWork:USERname](#)" on page 430
- "[":HARDcopy:NETWork:PASSword](#)" on page 428

## :HARDcopy:NETWork:APPLy

**N** (see [page 1334](#))

**Command Syntax** :HARDcopy:NETWork:APPLy

The :HARDcopy:NETWork:APPLy command applies the network printer settings and makes the printer connection.

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[":HARDcopy:NETWork:SLOT](#)" on page 429
- "[":HARDcopy:NETWork:ADDRess](#)" on page 425
- "[":HARDcopy:NETWork:DOMain](#)" on page 427
- "[":HARDcopy:NETWork:USERname](#)" on page 430
- "[":HARDcopy:NETWork:PASSword](#)" on page 428

## :HARDcopy:NETWork:DOMain

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:NETWork:DOMain <domain>`  
`<domain> ::= quoted ASCII string`

The :HARDcopy:NETWork:DOMain command sets the Windows network domain name.

The domain name setting is a common setting for both network printer slots.

**Query Syntax**    `:HARDcopy:NETWork:DOMain?`

The :HARDcopy:NETWork:DOMain? query returns the current Windows network domain name.

**Return Format**    `<domain><NL>`  
`<domain> ::= quoted ASCII string`

**See Also**

- "Introduction to [:HARDcopy Commands](#)" on page 418
- "[:HARDcopy:NETWork:SLOT](#)" on page 429
- "[:HARDcopy:NETWork:APPLy](#)" on page 426
- "[:HARDcopy:NETWork:ADDResS](#)" on page 425
- "[:HARDcopy:NETWork:USERname](#)" on page 430
- "[:HARDcopy:NETWork:PASSword](#)" on page 428

## :HARDcopy:NETWork:PASSword

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:NETWork:PASSword <password>`

`<password>` ::= quoted ASCII string

The :HARDcopy:NETWork:PASSword command sets the password for the specified Windows network domain and user name.

The password setting is a common setting for both network printer slots.

- See Also**
- "[Introduction to :HARDcopy Commands](#)" on page 418
  - "[:HARDcopy:NETWork:USERname](#)" on page 430
  - "[:HARDcopy:NETWork:DOMain](#)" on page 427
  - "[:HARDcopy:NETWork:SLOT](#)" on page 429
  - "[:HARDcopy:NETWork:APPLy](#)" on page 426
  - "[:HARDcopy:NETWork:ADDRess](#)" on page 425

## :HARDcopy:NETWork:SLOT

**N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:NETWork:SLOT <slot>`  
`<slot> ::= {NET0 | NET1}`

The :HARDcopy:NETWork:SLOT command selects the network printer slot used for the address and apply commands. There are two network printer slots to choose from.

**Query Syntax**    `:HARDcopy:NETWork:SLOT?`

The :HARDcopy:NETWork:SLOT? query returns the currently selected network printer slot.

**Return Format**    `<slot><NL>`  
`<slot> ::= {NET0 | NET1}`

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[":HARDcopy:NETWork:APPLy](#)" on page 426
- "[":HARDcopy:NETWork:ADDRess](#)" on page 425
- "[":HARDcopy:NETWork:DOMain](#)" on page 427
- "[":HARDcopy:NETWork:USERname](#)" on page 430
- "[":HARDcopy:NETWork:PASSword](#)" on page 428

**:HARDcopy:NETWork:USERname****N** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:NETWork:USERname <username>`  
`<username> ::= quoted ASCII string`

The :HARDcopy:NETWork:USERname command sets the user name to use when connecting to the Windows network domain.

The user name setting is a common setting for both network printer slots.

**Query Syntax**    `:HARDcopy:NETWork:USERname?`

The :HARDcopy:NETWork:USERname? query returns the currently set user name.

**Return Format**    `<username><NL>`  
`<username> ::= quoted ASCII string`

- See Also**
- "[Introduction to :HARDcopy Commands](#)" on page 418
  - "[":HARDcopy:NETWork:DOMain](#)" on page 427
  - "[":HARDcopy:NETWork:PASSword](#)" on page 428
  - "[":HARDcopy:NETWork:SLOT](#)" on page 429
  - "[":HARDcopy:NETWork:APPLy](#)" on page 426
  - "[":HARDcopy:NETWork:ADDRess](#)" on page 425

## :HARDcopy:PALETTE

**N** (see [page 1334](#))

**Command Syntax** :HARDcopy:PALETTE <palette>

<palette> ::= {COLOR | GRAYscale | NONE}

The :HARDcopy:PALETTE command sets the hardcopy palette color.

The oscilloscope's print driver cannot print color images to color laser printers, so the COLOR option is not available when connected to laser printers.

**Query Syntax** :HARDcopy:PALETTE?

The :HARDcopy:PALETTE? query returns the selected hardcopy palette color.

**Return Format** <palette><NL>

<palette> ::= {COL | GRAY | NONE}

- See Also**
- "[Introduction to :HARDcopy Commands](#)" on page 418
  - "[":HARDcopy:STARt](#)" on page 433
  - "[":HARDcopy:FACTors](#)" on page 421
  - "[":HARDcopy:LAYout](#)" on page 424
  - "[":HARDcopy:FFeed](#)" on page 422
  - "[":HARDcopy:INKSaver](#)" on page 423

**:HARDcopy:PRINter:LIST****N** (see [page 1334](#))**Query Syntax**    `:HARDcopy:PRINter:LIST?`

The `:HARDcopy:PRINter:LIST?` query returns a list of available printers. The list can be empty.

**Return Format**    `<list><NL>`

```
<list> ::= [<printer_spec>] ... [printer_spec]>  
<printer_spec> ::= "<index>,<active>,<name>;"  
<index> ::= integer index of printer  
<active> ::= {Y | N}  
<name> ::= name of printer (for example "DESKJET 950C")
```

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[":HARDcopy:APRinter](#)" on page 420
- "[":HARDcopy:STARt](#)" on page 433

## :HARDcopy:STARt

**N** (see [page 1334](#))

**Command Syntax** :HARDcopy:STARt

The :HARDcopy:STARt command starts a print job.

**See Also**

- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[:HARDcopy:APRINTER](#)" on page 420
- "[:HARDcopy:PRINTER:LIST](#)" on page 432
- "[:HARDcopy:FACTors](#)" on page 421
- "[:HARDcopy:FFEed](#)" on page 422
- "[:HARDcopy:INKSaver](#)" on page 423
- "[:HARDcopy:LAYOUT](#)" on page 424
- "[:HARDcopy:PALETTE](#)" on page 431



## 20 :LISTer Commands

**Table 101** :LISTer Commands Summary

Command	Query	Options and Query Returns
n/a	:LISTer:DATA? (see <a href="#">page 436</a> )	<binary_block> ::= comma-separated data with newlines at the end of each row
:LISTer:DISPlay {{OFF   0}   {SBUS1   ON   1}   {SBUS2   2}   ALL} (see <a href="#">page 437</a> )	:LISTer:DISPlay? (see <a href="#">page 437</a> )	{OFF   SBUS1   SBUS2   ALL}
:LISTer:REFerence <time_ref> (see <a href="#">page 438</a> )	:LISTer:REFerence? (see <a href="#">page 438</a> )	<time_ref> ::= {TRIGger   PREVIOUS}

**Introduction to :LISTer Commands** The LISTer subsystem is used to turn on/off the serial decode Lister display and return data from the Lister display.

**:LISTer:DATA****N** (see [page 1334](#))**Query Syntax** `:LISTer:DATA?`

The `:LISTer:DATA?` query returns the lister data.

**Return Format** `<binary block><NL>`

`<binary_block>` ::= comma-separated data with newlines at the end of each row

**See Also**

- ["Introduction to :LISTer Commands"](#) on page 435
- [":LISTer:DISPLAY"](#) on page 437
- ["Definite-Length Block Response Data"](#) on page 175

## :LISTer:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:LISTer:DISPlay <value>`

`<value> ::= {{OFF | 0} | {SBUS1 | ON | 1} | {SBUS2 | 2} | ALL}`

The :LISTer:DISPlay command configures which of the serial buses to display in the Lister, or whether the Lister is off. "ON" or "1" is the same as "SBUS1".

When set to "ALL", the decode information for different buses is interleaved in time.

Serial bus decode must be on before it can be displayed in the Lister.

**Query Syntax**    `:LISTer:DISPlay?`

The :LISTer:DISPlay? query returns the Lister display setting.

**Return Format**    `<value><NL>`

`<value> ::= {OFF | SBUS1 | SBUS2 | ALL}`

**See Also**

- "[Introduction to :LISTer Commands](#)" on page 435
- "[":SBUS<n>:DISPLAY](#)" on page 726
- "[":LISTer:DATA](#)" on page 436

**:LISTER:REFerence**

**N** (see [page 1334](#))

**Command Syntax**    `:LISTER:REFerence <time_ref>`  
`<time_ref> ::= {TRIGger | PREVIOUS}`

The :LISTER:REFerence command selects whether the time value for a Lister row is relative to the trigger or the previous Lister row.

**Query Syntax**    `:LISTER:REFerence?`

The :LISTER:REFerence? query returns the Lister time reference setting.

**Return Format**    `<time_ref><NL>`  
`<time_ref> ::= {TRIGger | PREVIOUS}`

**See Also**

- "[Introduction to :LISTER Commands](#)" on page 435
- "[":SBUS<n>:DISPLAY](#)" on page 726
- "[":LISTER:DATA](#)" on page 436
- "[":LISTER:DISPLAY](#)" on page 437

# 21 :MARKer Commands

Set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors). See "[Introduction to :MARKer Commands](#)" on page 441.

**Table 102**:MARKer Commands Summary

Command	Query	Options and Query Returns
n/a	:MARKer:DYDX? (see <a href="#">page 442</a> )	<return_value> ::= •Y/•X value in NR3 format
:MARKer:MODE <mode> (see <a href="#">page 443</a> )	:MARKer:MODE? (see <a href="#">page 443</a> )	<mode> ::= {OFF   MEASurement   MANual   WAVEform   BINARY   HEX}
:MARKer:X1:DISPLAY {{0   OFF}   {1   ON}} (see <a href="#">page 444</a> )	:MARKer:X1:DISPLAY? (see <a href="#">page 444</a> )	<setting> ::= {0   1}
:MARKer:X1Position <position>[suffix] (see <a href="#">page 445</a> )	:MARKer:X1Position? (see <a href="#">page 445</a> )	<position> ::= X1 cursor position value in NR3 format [suffix] ::= {s   ms   us   ns   ps   Hz   kHz   MHz} <return_value> ::= X1 cursor position value in NR3 format
:MARKer:X1Y1source <source> (see <a href="#">page 446</a> )	:MARKer:X1Y1source? (see <a href="#">page 446</a> )	<source> ::= {CHANnel<n>   FUNCTION<m>   FFT   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= <source>
:MARKer:X2:DISPLAY {{0   OFF}   {1   ON}} (see <a href="#">page 447</a> )	:MARKer:X2:DISPLAY? (see <a href="#">page 447</a> )	<setting> ::= {0   1}

**Table 102:** MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:X2Position <position>[suffix] (see <a href="#">page 448</a> )	:MARKer:X2Position? (see <a href="#">page 448</a> )	<position> ::= X2 cursor position value in NR3 format [suffix] ::= {s   ms   us   ns   ps   Hz   kHz   MHz} <return_value> ::= X2 cursor position value in NR3 format
:MARKer:X2Y2source <source> (see <a href="#">page 449</a> )	:MARKer:X2Y2source? (see <a href="#">page 449</a> )	<source> ::= {CHANnel<n>   FUNCtion<m>   FFT   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= <source>
n/a	:MARKer:XDELta? (see <a href="#">page 450</a> )	<return_value> ::= X cursors delta value in NR3 format
:MARKer:XUNits <mode> (see <a href="#">page 451</a> )	:MARKer:XUNits? (see <a href="#">page 451</a> )	<units> ::= {SEConds   HERTZ   DEGRees   PERCent}
:MARKer:XUNits:USE (see <a href="#">page 452</a> )	n/a	n/a
:MARKer:Y1:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 453</a> )	:MARKer:Y1:DISPLAY? (see <a href="#">page 453</a> )	<setting> ::= {0   1}
:MARKer:Y1Position <position>[suffix] (see <a href="#">page 454</a> )	:MARKer:Y1Position? (see <a href="#">page 454</a> )	<position> ::= Y1 cursor position value in NR3 format [suffix] ::= {V   mV   dB} <return_value> ::= Y1 cursor position value in NR3 format
:MARKer:Y2:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 455</a> )	:MARKer:Y2:DISPLAY? (see <a href="#">page 455</a> )	<setting> ::= {0   1}
:MARKer:Y2Position <position>[suffix] (see <a href="#">page 456</a> )	:MARKer:Y2Position? (see <a href="#">page 456</a> )	<position> ::= Y2 cursor position value in NR3 format [suffix] ::= {V   mV   dB} <return_value> ::= Y2 cursor position value in NR3 format
n/a	:MARKer:YDELta? (see <a href="#">page 457</a> )	<return_value> ::= Y cursors delta value in NR3 format

**Table 102:** MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:YUNits <mode> (see <a href="#">page 458</a> )	:MARKer:YUNits? (see <a href="#">page 458</a> )	<units> ::= {BASE   PERCent}
:MARKer:YUNits:USE (see <a href="#">page 459</a> )	n/a	n/a

**Introduction to :MARKer Commands** The MARKer subsystem commands set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors). You can set and query the marker mode and source, the position of the X and Y cursors, and query delta X and delta Y cursor values.

#### Reporting the Setup

Use :MARKer? to query setup information for the MARKer subsystem.

#### Return Format

The following is a sample response from the :MARKer? query. In this case, the query was issued following a \*RST and ":MARKer:MODE MANual" command.

```
:MARK:X1Y1 CHAN1;X2Y2 CHAN1;MODE MAN
```

**:MARKer:DYDX****N** (see [page 1334](#))**Query Syntax** `:MARKer:DYDX?`

The MARKer:DYDX? query returns the cursor ΔY/ΔX value.

X cursor units are set by the :MARKer:XUNits command.

**NOTE**

If the front-panel cursors are off, the marker position values are not defined. Make sure to set :MARKer:MODE to MANual or WAveform to put the cursors in the front-panel Normal mode.

**Return Format** `<value><NL>`

`<value>` ::= •Y/•X value in NR3 format.

**See Also**

- ["Introduction to :MARKer Commands"](#) on page 441
- [":MARKer:MODE"](#) on page 443
- [":MARKer:X1Position"](#) on page 445
- [":MARKer:X2Position"](#) on page 448
- [":MARKer:X1Y1source"](#) on page 446
- [":MARKer:X2Y2source"](#) on page 449
- [":MARKer:XUNits"](#) on page 451

## :MARKer:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:MODE <mode>`

`<mode> ::= {OFF | MEASurement | MANual | WAVEform | BINary | HEX}`

The :MARKer:MODE command sets the cursors mode:

- OFF – removes the cursor information from the display.
- MANual – enables manual placement of the X and Y cursors.

If the front-panel cursors are off, or are set to the front-panel Hex or Binary mode, setting :MARKer:MODE MANual will put the cursors in the front-panel Normal mode.

- MEASurement – cursors track the most recent measurement.

Setting the mode to MEASurement sets the marker sources (:MARKer:X1Y1source and :MARKer:X2Y2source) to the measurement source (:MEASure:SOURce). Setting the measurement source remotely always sets the marker sources.

- WAVEform – the Y1 cursor tracks the voltage value at the X1 cursor of the waveform specified by the X1Y1source, and the Y2 cursor does the same for the X2 cursor and its X2Y2source.
- BINary – logic levels of displayed waveforms at the current X1 and X2 cursor positions are displayed in the Cursor sidebar dialog in binary.
- HEX – logic levels of displayed waveforms at the current X1 and X2 cursor positions are displayed in the Cursor sidebar dialog in hexadecimal.

**Query Syntax**    `:MARKer:MODE?`

The :MARKer:MODE? query returns the current cursors mode.

**Return Format**    `<mode><NL>`

`<mode> ::= {OFF | MEAS | MAN | WAV | BIN | HEX}`

- See Also**
- "[Introduction to :MARKer Commands](#)" on page 441
  - "[":MARKer:X1Y1source](#)" on page 446
  - "[":MARKer:X2Y2source](#)" on page 449
  - "[":MEASure:SOURce](#)" on page 517
  - "[":MARKer:X1Position](#)" on page 445
  - "[":MARKer:X2Position](#)" on page 448
  - "[":MARKer:Y1Position](#)" on page 454
  - "[":MARKer:Y2Position](#)" on page 456

## :MARKer:X1:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:X1:DISPlay {{0 | OFF} | {1 | ON}}`

The :MARKer:X1:DISPlay command specifies whether the X1 cursor is displayed.

**Query Syntax**    `:MARKer:X1:DISPlay?`

The :MARKer:X1:DISPlay? query returns the X1 cursor display setting.

**Return Format**    `<setting><NL>`

`<setting> ::= {0 | 1}`

**See Also**    • [":MARKer:X1:DISPlay"](#) on page 444

## :MARKer:X1Position

**N** (see [page 1334](#))

- Command Syntax**
- ```
:MARKer:X1Position <position> [suffix]
<position> ::= X1 cursor position in NR3 format
<suffix> ::= {s | ms | us | ns | ps | Hz | kHz | MHz}
```
- The :MARKer:X1Position command:
- Sets :MARKer:MODE to MANual if it is not currently set to WAVerform (see "[:MARKer:MODE](#)" on page 443).
  - Sets the X1 cursor position to the specified value.
- X cursor units are set by the :MARKer:XUNits command.

- Query Syntax**

```
:MARKer:X1Position?
```

The :MARKer:X1Position? query returns the current X1 cursor position. This is functionally equivalent to the obsolete :MEASure:TSTArt command/query.

**NOTE**

If the front-panel cursors are off, the marker position values are not defined and an error is generated. Make sure to set :MARKer:MODE to MANual or WAVerform to put the cursors in the front-panel Normal mode.

- 
- |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Return Format</b> | <pre>&lt;position&gt;&lt;NL&gt; &lt;position&gt; ::= X1 cursor position in NR3 format</pre>                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>See Also</b>      | <ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :MARKer Commands</a>" on page 441</li> <li>• "<a href="#">:MARKer:MODE</a>" on page 443</li> <li>• "<a href="#">:MARKer:X2Position</a>" on page 448</li> <li>• "<a href="#">:MARKer:X1Y1source</a>" on page 446</li> <li>• "<a href="#">:MARKer:X2Y2source</a>" on page 449</li> <li>• "<a href="#">:MARKer:XUNits</a>" on page 451</li> <li>• "<a href="#">:MEASure:TSTArt</a>" on page 1266</li> </ul> |

## :MARKer:X1Y1source

**N** (see [page 1334](#))

**Command Syntax**

```
:MARKer:X1Y1source <source>
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MARKer:X1Y1source command sets the source for the cursors. The channel you specify must be enabled for cursors to be displayed. If the channel or function is not on, an error message is issued.

If the marker mode is not currently WAVEform (see "[:MARKer:MODE](#)" on page 443):

- Sending a :MARKer:X1Y1source command will put the cursors in the MANual mode.
- Setting the source for one pair of markers (for example, X1Y1) sets the source for the other (for example, X2Y2).

If the marker mode is currently WAVEform, the X1Y1 source can be set separate from the X2Y2 source.

If :MARKer:MODE is set to OFF or MANual, setting :MEASure:SOURce to CHANnel<n>, FUNCtion, MATH, or WMEMory<r> will also set :MARKer:X1Y1source and :MARKer:X2Y2source to this value.

### NOTE

MATH is an alias for FUNCtion. The query will return FUNC if the source is FUNCtion or MATH.

**Query Syntax**

:MARKer:X1Y1source?

The :MARKer:X1Y1source? query returns the current source for the cursors. If all channels are off or if :MARKer:MODE is set to OFF, the query returns NONE.

**Return Format**

```
<source><NL>
<source> ::= {CHAN<n> | FUNC | WMEM<r> | NONE}
```

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[:MARKer:MODE](#)" on page 443
- "[:MARKer:X2Y2source](#)" on page 449
- "[:MEASure:SOURce](#)" on page 517

## :MARKer:X2:DISPLAY

**N** (see [page 1334](#))

**Command Syntax** :MARKer:X2:DISPLAY {{0 | OFF} | {1 | ON}}

The :MARKer:X2:DISPLAY command specifies whether the X2 cursor is displayed.

**Query Syntax** :MARKer:X2:DISPLAY?

The :MARKer:X2:DISPLAY? query returns the X2 cursor display setting.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

**See Also** • [":MARKer:X2:DISPLAY"](#) on page 447

## :MARKer:X2Position

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:X2Position <position> [suffix]`

`<position> ::= X2 cursor position in NR3 format`

`<suffix> ::= {s | ms | us | ns | ps | Hz | kHz | MHz}`

The :MARKer:X2Position command:

- Sets :MARKer:MODE to MANual if it is not currently set to WAVerform (see "[:MARKer:MODE](#)" on page 443).
- Sets the X2 cursor position to the specified value.

X cursor units are set by the :MARKer:XUNits command.

**Query Syntax**    `:MARKer:X2Position?`

The :MARKer:X2Position? query returns current X2 cursor position. This is functionally equivalent to the obsolete :MEASure:TSTOP command/query.

### NOTE

If the front-panel cursors are off, the marker position values are not defined and an error is generated. Make sure to set :MARKer:MODE to MANual or WAVerform to put the cursors in the front-panel Normal mode.

**Return Format**    `<position><NL>`

`<position> ::= X2 cursor position in NR3 format`

**See Also**    ["Introduction to :MARKer Commands"](#) on page 441

- "[:MARKer:MODE](#)" on page 443
- "[:MARKer:X1Position](#)" on page 445
- "[:MARKer:X2Y2source](#)" on page 449
- "[:MARKer:XUNits](#)" on page 451
- "[:MEASure:TSTOP](#)" on page 1267

## :MARKer:X2Y2source

**N** (see [page 1334](#))

### Command Syntax

```
:MARKer:X2Y2source <source>
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MARKer:X2Y2source command sets the source for the cursors. The channel you specify must be enabled for cursors to be displayed. If the channel or function is not on, an error message is issued.

If the marker mode is not currently WAVEform (see "[:MARKer:MODE](#)" on page 443):

- Sending a :MARKer:X2Y2source command will put the cursors in the MANual mode.
- Setting the source for one pair of markers (for example, X2Y2) sets the source for the other (for example, X1Y1).

If the marker mode is currently WAVEform, the X2Y2 source can be set separate from the X1Y1 source.

If :MARKer:MODE is set to OFF or MANual, setting :MEASure:SOURce to CHANnel<n>, FUNCtion, MATH, or WMEMory<r> will also set :MARKer:X1Y1source and :MARKer:X2Y2source to this value.

### NOTE

MATH is an alias for FUNCtion. The query will return FUNC if the source is FUNCtion or MATH.

### Query Syntax

```
:MARKer:X2Y2source?
```

The :MARKer:X2Y2source? query returns the current source for the cursors. If all channels are off or if :MARKer:MODE is set to OFF, the query returns NONE.

### Return Format

```
<source><NL>
<source> ::= {CHAN<n> | FUNC | WMEM<r> | NONE}
```

### See Also

- "[Introduction to :MARKer Commands](#)" on page 441
- "[:MARKer:MODE](#)" on page 443
- "[:MARKer:X1Y1source](#)" on page 446
- "[:MEASure:SOURce](#)" on page 517

## :MARKer:XDELta

**N** (see [page 1334](#))

**Query Syntax** `:MARKer:XDELta?`

The MARKer:XDELta? query returns the value difference between the current X1 and X2 cursor positions.

Xdelta = (Value at X2 cursor) - (Value at X1 cursor)

X cursor units are set by the :MARKer:XUNits command.

### NOTE

If the front-panel cursors are off, the marker position values are not defined. Make sure to set :MARKer:MODE to MANual or WAveform to put the cursors in the front-panel Normal mode.

**Return Format** `<value><NL>`

`<value>` ::= difference value in NR3 format.

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[":MARKer:MODE](#)" on page 443
- "[":MARKer:X1Position](#)" on page 445
- "[":MARKer:X2Position](#)" on page 448
- "[":MARKer:X1Y1source](#)" on page 446
- "[":MARKer:X2Y2source](#)" on page 449
- "[":MARKer:XUNits](#)" on page 451

## :MARKer:XUNits

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:XUNits <units>`

`<units> ::= {SEConds | HERTz | DEGRees | PERCent}`

The :MARKer:XUNits command sets the X cursors units:

- SEConds – for making time measurements.
- HERTz – for making frequency measurements.
- DEGRees – for making phase measurements. Use the :MARKer:XUNits:USE command to set the current X1 location as 0 degrees and the current X2 location as 360 degrees.
- PERCent – for making ratio measurements. Use the :MARKer:XUNits:USE command to set the current X1 location as 0 percent and the current X2 location as 100 percent.

Changing X units affects the input and output values of the :MARKer:X1Position, :MARKer:X2Position, and :MARKer:XDELta commands/queries.

**Query Syntax**    `:MARKer:XUNits?`

The :MARKer:XUNits? query returns the current X cursors units.

**Return Format**    `<units><NL>`

`<units> ::= {SEC | HERT | DEGR | PERC}`

- See Also**
- "[Introduction to :MARKer Commands](#)" on page 441
  - "[:MARKer:XUNits:USE](#)" on page 452
  - "[:MARKer:X1Y1source](#)" on page 446
  - "[:MARKer:X2Y2source](#)" on page 449
  - "[:MEASure:SOURce](#)" on page 517
  - "[:MARKer:X1Position](#)" on page 445
  - "[:MARKer:X2Position](#)" on page 448

## :MARKer:XUNits:USE

**N** (see [page 1334](#))

**Command Syntax** `:MARKer:XUNits:USE`

When DEGRees is selected for :MARKer:XUNits, the :MARKer:XUNits:USE command sets the current X1 location as 0 degrees and the current X2 location as 360 degrees.

When PERCent is selected for :MARKer:XUNits, the :MARKer:XUNits:USE command sets the current X1 location as 0 percent and the current X2 location as 100 percent.

Once the 0 and 360 degree or 0 and 100 percent locations are set, inputs to and outputs from the :MARKer:X1Position, :MARKer:X2Position, and :MARKer:XDELta commands/queries are relative to the set locations.

- See Also**
- ["Introduction to :MARKer Commands"](#) on page 441
  - [":MARKer:XUNits"](#) on page 451
  - [":MARKer:X1Y1source"](#) on page 446
  - [":MARKer:X2Y2source"](#) on page 449
  - [":MEASure:SOURce"](#) on page 517
  - [":MARKer:X1Position"](#) on page 445
  - [":MARKer:X2Position"](#) on page 448
  - [":MARKer:XDELta"](#) on page 450

**:MARKer:Y1:DISPlay****N** (see [page 1334](#))**Command Syntax** :MARKer:Y1:DISPlay {{0 | OFF} | {1 | ON}}

The :MARKer:Y1:DISPlay command specifies whether the Y1 cursor is displayed.

**Query Syntax** :MARKer:Y1:DISPlay?

The :MARKer:Y1:DISPlay? query returns the Y1 cursor display setting.

**Return Format** <setting><NL>

&lt;setting&gt; ::= {0 | 1}

**See Also** • [":MARKer:Y1:DISPlay"](#) on page 453

## :MARKer:Y1Position

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:Y1Position <position> [<suffix>]`

`<position> ::= Y1 cursor position in NR3 format`

`<suffix> ::= {mV | V | dB}`

If the :MARKer:MODE is not currently set to WAVEform (see "[:MARKer:MODE](#)" on page 443), the :MARKer:Y1Position command:

- Sets :MARKer:MODE to MANual.
- Sets the Y1 cursor position to the specified value.

Y cursor units are set by the :MARKer:YUNits command.

When the :MARKer:MODE is set to WAVEform, Y positions cannot be set.

**Query Syntax**    `:MARKer:Y1Position?`

The :MARKer:Y1Position? query returns current Y1 cursor position. This is functionally equivalent to the obsolete :MEASure:VSTArt command/query.

### NOTE

If the front-panel cursors are off or are set to Binary or Hex Mode, the marker position values are not defined and an error is generated. Make sure to set :MARKer:MODE to MANual or WAVEform to put the cursors in the front-panel Normal mode.

**Return Format**    `<position><NL>`

`<position> ::= Y1 cursor position in NR3 format`

**See Also**    ["Introduction to :MARKer Commands"](#) on page 441

- [":MARKer:MODE"](#) on page 443
- [":MARKer:X1Y1source"](#) on page 446
- [":MARKer:X2Y2source"](#) on page 449
- [":MARKer:Y2Position"](#) on page 456
- [":MARKer:YUNits"](#) on page 458
- [":MEASure:VSTArt"](#) on page 1271

**:MARKer:Y2:DISPlay****N** (see [page 1334](#))**Command Syntax**    `:MARKer:Y2:DISPlay {{0 | OFF} | {1 | ON}}`

The :MARKer:Y2:DISPlay command specifies whether the Y2 cursor is displayed.

**Query Syntax**    `:MARKer:Y2:DISPlay?`

The :MARKer:Y2:DISPlay? query returns the Y2 cursor display setting.

**Return Format**    `<setting><NL>``<setting> ::= {0 | 1}`**See Also**    • [":MARKer:Y2:DISPlay"](#) on page 455

## :MARKer:Y2Position

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:Y2Position <position> [suffix]`

`<position> ::= Y2 cursor position in NR3 format`

`<suffix> ::= {mV | v | dB}`

If the :MARKer:MODE is not currently set to WAVEform (see "[:MARKer:MODE](#)" on page 443), the :MARKer:Y1Position command:

- Sets :MARKer:MODE to MANual.
- Sets the Y2 cursor position to the specified value.

Y cursor units are set by the :MARKer:YUNits command.

When the :MARKer:MODE is set to WAVEform, Y positions cannot be set.

**Query Syntax**    `:MARKer:Y2Position?`

The :MARKer:Y2Position? query returns current Y2 cursor position. This is functionally equivalent to the obsolete :MEASure:VSTOP command/query.

### NOTE

If the front-panel cursors are off or are set to Binary or Hex Mode, the marker position values are not defined and an error is generated. Make sure to set :MARKer:MODE to MANual or WAVEform to put the cursors in the front-panel Normal mode.

**Return Format**    `<position><NL>`

`<position> ::= Y2 cursor position in NR3 format`

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[:MARKer:MODE](#)" on page 443
- "[:MARKer:X1Y1source](#)" on page 446
- "[:MARKer:X2Y2source](#)" on page 449
- "[:MARKer:Y1Position](#)" on page 454
- "[:MARKer:YUNits](#)" on page 458
- "[:MEASure:VSTOP](#)" on page 1272

## :MARKer:YDELta

**N** (see [page 1334](#))

**Query Syntax** :MARKer:YDELta?

The :MARKer:YDELta? query returns the value difference between the current Y1 and Y2 cursor positions.

Ydelta = (Value at Y2 cursor) - (Value at Y1 cursor)

### NOTE

If the front-panel cursors are off or are set to Binary or Hex Mode, the marker position values are not defined. Make sure to set :MARKer:MODE to MANual or WAVeform to put the cursors in the front-panel Normal mode.

Y cursor units are set by the :MARKer:YUNits command.

**Return Format** <value><NL>

<value> ::= difference value in NR3 format

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[":MARKer:MODE](#)" on page 443
- "[":MARKer:X1Y1source](#)" on page 446
- "[":MARKer:X2Y2source](#)" on page 449
- "[":MARKer:Y1Position](#)" on page 454
- "[":MARKer:Y2Position](#)" on page 456
- "[":MARKer:YUNits](#)" on page 458

## :MARKer:YUNits

**N** (see [page 1334](#))

**Command Syntax**    `:MARKer:YUNits <units>`

`<units> ::= {BASE | PERCent}`

The :MARKer:YUNits command sets the Y cursors units:

- BASE – for making measurements in the units associated with the cursors source.
- PERCent – for making ratio measurements. Use the :MARKer:YUNits:USE command to set the current Y1 location as 0 percent and the current Y2 location as 100 percent.

Changing Y units affects the input and output values of the :MARKer:Y1Position, :MARKer:Y2Position, and :MARKer:YDELta commands/queries.

**Query Syntax**    `:MARKer:YUNits?`

The :MARKer:YUNits? query returns the current Y cursors units.

**Return Format**    `<units><NL>`

`<units> ::= {BASE | PERC}`

- See Also**
- "[Introduction to :MARKer Commands](#)" on page 441
  - "[":MARKer:YUNits:USE](#)" on page 459
  - "[":MARKer:X1Y1source](#)" on page 446
  - "[":MARKer:X2Y2source](#)" on page 449
  - "[":MEASure:SOURce](#)" on page 517
  - "[":MARKer:Y1Position](#)" on page 454
  - "[":MARKer:Y2Position](#)" on page 456
  - "[":MARKer:YDELta](#)" on page 457

## :MARKer:YUNits:USE

**N** (see [page 1334](#))

**Command Syntax** `:MARKer:YUNits:USE`

When PERCent is selected for :MARKer:YUNits, the :MARKer:YUNits:USE command sets the current Y1 location as 0 percent and the current Y2 location as 100 percent.

Once the 0 and 100 percent locations are set, inputs to and outputs from the :MARKer:Y1Position, :MARKer:Y2Position, and :MARKer:YDELta commands/queries are relative to the set locations.

**See Also**

- ["Introduction to :MARKer Commands"](#) on page 441
- [":MARKer:YUNits"](#) on page 458
- [":MARKer:X1Y1source"](#) on page 446
- [":MARKer:X2Y2source"](#) on page 449
- [":MEASure:SOURce"](#) on page 517
- [":MARKer:Y1Position"](#) on page 454
- [":MARKer:Y2Position"](#) on page 456
- [":MARKer:YDELta"](#) on page 457



## 22 :MEASure Commands

Select automatic measurements to be made and control time markers. See "[Introduction to :MEASure Commands](#)" on page 476.

**Table 103**:MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:ALL (see <a href="#">page 478</a> )	n/a	n/a
:MEASure:AREA [ <i>&lt;interval&gt;</i> ] [, <i>&lt;source&gt;</i> ] (see <a href="#">page 479</a> )	:MEASure:AREA? [ <i>&lt;interval&gt;</i> ] [, <i>&lt;source&gt;</i> ] (see <a href="#">page 479</a> )	<i>&lt;interval&gt;</i> ::= {CYCLE   DISPLAY} <i>&lt;source&gt;</i> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <i>&lt;n&gt;</i> ::= 1 to (# analog channels) in NR1 format <i>&lt;m&gt;</i> ::= 1 to (# math functions) in NR1 format <i>&lt;r&gt;</i> ::= 1 to (# ref waveforms) in NR1 format <i>&lt;return_value&gt;</i> ::= area in volt-seconds, NR3 format
:MEASure:BRATE [ <i>&lt;source&gt;</i> ] (see <a href="#">page 480</a> )	:MEASure:BRATE? [ <i>&lt;source&gt;</i> ] (see <a href="#">page 480</a> )	<i>&lt;source&gt;</i> ::= { <i>&lt;digital channels&gt;</i>   CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <i>&lt;digital channels&gt;</i> ::= DIGITAL<d> for the MSO models <i>&lt;d&gt;</i> ::= 0 to (# digital channels - 1) in NR1 format <i>&lt;n&gt;</i> ::= 1 to (# of analog channels) in NR1 format <i>&lt;m&gt;</i> ::= 1 to (# math functions) in NR1 format <i>&lt;r&gt;</i> ::= 1 to (# ref waveforms) in NR1 format <i>&lt;return_value&gt;</i> ::= bit rate in Hz, NR3 format

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:BWIDth [<source>] (see <a href="#">page 481</a> )	:MEASure:BWIDth? [<source>] (see <a href="#">page 481</a> )	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= burst width in seconds, NR3 format
:MEASure:CLEar (see <a href="#">page 482</a> )	n/a	n/a
:MEASure:COUNTER [<source>] (see <a href="#">page 483</a> )	:MEASure:COUNTER? [<source>] (see <a href="#">page 483</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   EXTernal} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= counter frequency in Hertz in NR3 format
:MEASure:DEFine DEDelay, <delay spec> (see <a href="#">page 485</a> )	:MEASure:DEFine? DEDelay (see <a href="#">page 487</a> )	<delay spec> ::= <edge_spec1>, <edge_spec2> edge_spec1 ::= [<slope>] <occurrence> edge_spec2 ::= [<slope>] <occurrence> <slope> ::= {+   -} <occurrence> ::= integer
:MEASure:DEFine THresholds, <threshold spec> (see <a href="#">page 485</a> )	:MEASure:DEFine? THresholds (see <a href="#">page 487</a> )	<threshold spec> ::= {STANDARD}   {<threshold mode>, <upper>, <middle>, <lower>} <threshold mode> ::= {PERCent   ABSolute}

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DElay [<source1>] [,<source2>] (see <a href="#">page 488</a> )	:MEASure:DElay? [<source1>] [,<source2>] (see <a href="#">page 488</a> )	<source1,2> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUAL:CHARGE [<interval>] [,<source1>][,<source2>] (see <a href="#">page 490</a> )	:MEASure:DUAL:CHARge? [<interval>] [,<source1>][,<source2>] (see <a href="#">page 490</a> )	<interval> ::= {CYCLE   DISPLAY} <source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= area in Amp-hours, NR3 format
:MEASure:DUAL:VAMPtitude [<source1>][,<source2>] (see <a href="#">page 491</a> )	:MEASure:DUAL:VAMPtitude? [<source1>][,<source2>] (see <a href="#">page 491</a> )	<source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the amplitude of the selected waveform in volts in NR3 format
:MEASure:DUAL:VAverage [<interval>] [,<source1>][,<source2>] (see <a href="#">page 492</a> )	:MEASure:DUAL:VAverage? [<interval>] [,<source1>][,<source2>] (see <a href="#">page 492</a> )	<interval> ::= {CYCLE   DISPLAY} <source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= calculated average voltage in NR3 format
:MEASure:DUAL:VBASe [<source1>][,<source2>] (see <a href="#">page 493</a> )	:MEASure:DUAL:VBASe? [<source1>][,<source2>] (see <a href="#">page 493</a> )	<source1>,<source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <base_voltage> ::= voltage at the base of the selected waveform in NR3 format

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DUAL:VPP [<source1>] [,<source2>] (see <a href="#">page 494</a> )	:MEASure:DUAL:VPP? [<source1>] [,<source2>] (see <a href="#">page 494</a> )	<source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= voltage peak-to-peak of the selected waveform in NR3 format
:MEASure:DUAL:VRMS [<interval>] [,<type>] [,<source1>] [,<source2>] (see <a href="#">page 495</a> )	:MEASure:DUAL:VRMS? [<interval>] [,<type>] [,<source1>] [,<source2>] (see <a href="#">page 495</a> )	<interval> ::= {CYCLE   DISPLAY} <type> ::= {AC   DC} <source1>, <source2> ::= CHANnel<n> with N2820A probe connected <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= calculated RMS voltage in NR3 format
:MEASure:DUTYcycle [<source>] (see <a href="#">page 496</a> )	:MEASure:DUTYcycle? [<source>] (see <a href="#">page 496</a> )	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   FUNCtion<m>   MATH<m>   WMEMory<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FALLtime [<source>] (see page 497)	:MEASure:FALLtime? [<source>] (see page 497)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= time in seconds between the lower and upper thresholds in NR3 format</p>
:MEASure:FREQuency [<source>] (see page 498)	:MEASure:FREQuency? [<source>] (see page 498)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= frequency in Hertz in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NDUTy [<source>] (see page 499)	:MEASure:NDUTy? [<source>] (see page 499)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= ratio of negative pulse width to period in NR3 format</p>
:MEASure:NEDGes [<source>] (see page 500)	:MEASure:NEDGes? [<source>] (see page 500)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the falling edge count in NR3 format</p>
:MEASure:NPULses [<source>] (see page 501)	:MEASure:NPULses? [<source>] (see page 501)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the falling pulse count in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NWIDth [<source>] (see page 502)	:MEASure:NWIDth? [<source>] (see page 502)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= negative pulse width in seconds-NR3 format</p>
:MEASure:OVERshoot [<source>] (see page 503)	:MEASure:OVERshoot? [<source>] (see page 503)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the percent of the overshoot of the selected waveform in NR3 format</p>
:MEASure:PEDGes [<source>] (see page 505)	:MEASure:PEDGes? [<source>] (see page 505)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the rising edge count in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PERiod [<source>] (see page 506)	:MEASure:PERiod? [<source>] (see page 506)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= waveform period in seconds in NR3 format</p>
:MEASure:PHASE [<source1>] [,<source2>] (see page 507)	:MEASure:PHASE? [<source1>] [,<source2>] (see page 507)	<p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the phase angle value in degrees in NR3 format</p>
:MEASure:PPULses [<source>] (see page 508)	:MEASure:PPULses? [<source>] (see page 508)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the rising pulse count in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PREShoot [<source>] (see page 509)	:MEASure:PREShoot? [<source>] (see page 509)	<pre>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;} &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format &lt;return_value&gt; ::= the percent of preshoot of the selected waveform in NR3 format</pre>
:MEASure:PWIDth [<source>] (see page 510)	:MEASure:PWIDth? [<source>] (see page 510)	<pre>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;} for DSO models &lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;} for MSO models &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format &lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format &lt;return_value&gt; ::= width of positive pulse in seconds in NR3 format</pre>
n/a	:MEASure:RESults? <result_list> (see page 511)	<pre>&lt;result_list&gt; ::= comma-separated list of measurement results</pre>
:MEASure:RISetime [<source>] (see page 514)	:MEASure:RISetime? [<source>] (see page 514)	<pre>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;} &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format &lt;return_value&gt; ::= rise time in seconds in NR3 format</pre>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:SDEViation [ <source/> ] (see page 515)	:MEASure:SDEViation? [ <source/> ] (see page 515)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= calculated std deviation in NR3 format</p>
:MEASure:SHOW {{0   OFF}   {1   ON}} (see page 516)	:MEASure:SHOW? (see page 516)	{0   1}
:MEASure:SOURce <source1> [,<source2>] (see page 517)	:MEASure:SOURce? (see page 517)	<p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;   EXTERNAL} for DSO models</p> <p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMMemory&lt;r&gt;   EXTERNAL} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</p> <p>&lt;return_value&gt; ::= {&lt;source&gt;   NONE}</p>
:MEASure:STATistics <type> (see page 519)	:MEASure:STATistics? (see page 519)	<p>&lt;type&gt; ::= {{ON   1}   CURRent   MEAN   MINimum   MAXimum   STDDev   COUNT}</p> <p>ON ::= all statistics returned</p>
:MEASure:STATistics:D ISPlay {{0   OFF}   {1   ON}} (see page 520)	:MEASure:STATistics:D ISPlay? (see page 520)	{0   1}
:MEASure:STATistics:I NCrement (see page 521)	n/a	n/a

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:STATistics:M CCount <setting> (see <a href="#">page 522</a> )	:MEASure:STATistics:M CCount? (see <a href="#">page 522</a> )	<setting> ::= {INFinite   <count>} <count> ::= 2 to 2000 in NR1 format
:MEASure:STATistics:R ESet (see <a href="#">page 523</a> )	n/a	n/a
:MEASure:STATistics:R SDeviation {{0   OFF}   {1   ON}} (see <a href="#">page 524</a> )	:MEASure:STATistics:R SDeviation? (see <a href="#">page 524</a> )	{0   1}
n/a	:MEASure:TEDGE? <slope><occurrence>[, <source>] (see <a href="#">page 525</a> )	<slope> ::= direction of the waveform <occurrence> ::= the transition to be reported <source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   FUNCtion<m>   MATH<m>   WMEMory<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= time in seconds of the specified transition

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:TValue? <value>, [<>slope>]<occurrence> [, <source>] (see page 527)	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} for DSO models <source> ::= {CHANnel<n>   DIGital<d>   FUNCtion<m>   MATH<m>   WMEMory<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= time in seconds of specified voltage crossing in NR3 format
:MEASure:VAMplitude [<source>] (see page 529)	:MEASure:VAMplitude? [<source>] (see page 529)	<source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the amplitude of the selected waveform in volts in NR3 format

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAverage [<interval>] [,<source>] (see page 530)	:MEASure:VAverage? [<interval>] [,<source>] (see page 530)	<p>&lt;interval&gt; ::= {CYCLE   DISPLAY}</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= calculated average voltage in NR3 format</p>
:MEASure:VBASE [<source>] (see page 531)	:MEASure:VBASE? [<source>] (see page 531)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;base_voltage&gt; ::= voltage at the base of the selected waveform in NR3 format</p>
:MEASure:VMAX [<source>] (see page 532)	:MEASure:VMAX? [<source>] (see page 532)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMORY&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= maximum voltage of the selected waveform in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VMIN [<source>] (see page 533)	:MEASure:VMIN? [<source>] (see page 533)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= minimum voltage of the selected waveform in NR3 format</p>
:MEASure:VPP [<source>] (see page 534)	:MEASure:VPP? [<source>] (see page 534)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= voltage peak-to-peak of the selected waveform in NR3 format</p>
:MEASure:VRATio [<interval>] [, <source 1>] [, <source2>] (see page 535)	:MEASure:VRATio? [<interval>] [, <source 1>] [, <source2>] (see page 535)	<p>&lt;interval&gt; ::= {CYCLE   DISPLAY}</p> <p>&lt;source1,2&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the ratio value in dB in NR3 format</p>

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VRMS [<interval>] [,<type>] [,<source>] (see <a href="#">page 536</a> )	:MEASure:VRMS? [<interval>] [,<type>] [,<source>] (see <a href="#">page 536</a> )	<interval> ::= {CYCLE   DISPLAY} <type> ::= {AC   DC} <source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= calculated dc RMS voltage in NR3 format
n/a	:MEASure:VTIMe? <vtimetime>[,<source>] (see <a href="#">page 537</a> )	<vtimetime> ::= displayed time from trigger in seconds in NR3 format <source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} for DSO models <source> ::= {CHANNEL<n>   DIGITAL<d>   FUNCTION<m>   MATH<m>   WMEMORY<r>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= voltage at the specified time in NR3 format
:MEASure:VTOP [<source>] (see <a href="#">page 538</a> )	:MEASure:VTOP? [<source>] (see <a href="#">page 538</a> )	<source> ::= {CHANNEL<n>   FUNCTION<m>   MATH<m>   WMEMORY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= voltage at the top of the waveform in NR3 format
:MEASure:WINDOW <type> (see <a href="#">page 539</a> )	:MEASure:WINDOW? (see <a href="#">page 539</a> )	<type> ::= {MAIN   ZOOM   AUTO   GATE}

**Table 103:** MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:XMAX [<source>] (see page 540)	:MEASure:XMAX? [<source>] (see page 540)	<pre>&lt;source&gt; ::= {CHANnel&lt;n&gt;    FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;    WMEMory&lt;r&gt;} &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format &lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</pre>
:MEASure:XMIN [<source>] (see page 541)	:MEASure:XMIN? [<source>] (see page 541)	<pre>&lt;source&gt; ::= {CHANnel&lt;n&gt;    FUNCTION&lt;m&gt;   FFT   MATH&lt;m&gt;    WMEMory&lt;r&gt;} &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format &lt;return_value&gt; ::= horizontal value of the minimum in NR3 format</pre>

## Introduction to :MEASure Commands

The commands in the MEASure subsystem are used to make parametric measurements on displayed waveforms.

### Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the oscilloscope screen.

Measurement Type	Portion of waveform that must be displayed
period, duty cycle, or frequency	at least one complete cycle
pulse width	the entire pulse
rise time	rising edge, top and bottom of pulse
fall time	falling edge, top and bottom of pulse

### Measurement Error

If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned for that measurement.

### Making Measurements

If more than one waveform, edge, or pulse is displayed, time measurements are made on the portion of the displayed waveform closest to the trigger reference (left, center, or right).

When making measurements in the zoomed (delayed) time base mode (:TIMEbase:MODE WINDow), the oscilloscope will attempt to make the measurement inside the zoomed sweep window. If the measurement is an average and there are not three edges, the oscilloscope will revert to the mode of making the measurement at the start of the main sweep.

When the command form is used, the measurement result is displayed on the instrument. When the query form of these measurements is used, the measurement is made one time, and the measurement result is returned over the bus.

Measurements are made on the displayed waveforms specified by the :MEASure:SOURce command. The MATH source is an alias for the FUNCtion source.

Not all measurements are available on the digital channels or FFT (Fast Fourier Transform).

### Reporting the Setup

Use the :MEASure? query to obtain setup information for the MEASure subsystem. (Currently, this is only :MEASure:SOURce.)

### Return Format

The following is a sample response from the :MEASure? query. In this case, the query was issued following a \*RST command.

```
:MEAS:SOUR CHAN1,CHAN2;STAT ON
```

## :MEASure:ALL

 (see [page 1334](#))

**Command Syntax**    `:MEASure:ALL`

This command installs a Snapshot All measurement on the screen.

**See Also**

- ["Introduction to :MEASure Commands"](#) on page 476

## :MEASure:AREa

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:MEASure:AREa [&lt;interval&gt; [,&lt;source&gt;]</code>
	<code>&lt;interval&gt; ::= {CYCLE   DISPLAY}</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMORY&lt;r&gt;}</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</code>
	<code>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</code>
	The :MEASure:AREa command installs an area measurement on screen. Area measurements show the area between the waveform and the ground level.
	The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

<b>Query Syntax</b>	<code>:MEASure:AREa? [&lt;interval&gt; [,&lt;source&gt;]</code>
	The :MEASure:AREa? query measures and returns the area value.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code>
	<code>&lt;value&gt; ::= the area value in volt-seconds in NR3 format</code>

- See Also**
- "[Introduction to :MEASure Commands](#)" on page 476
  - "[":MEASure:SOURce](#)" on page 517

## :MEASure:BRATe

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:BRATe [<source>]`

```
<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMory<r>}

<digital channels> ::= DIGItal<d> for the MSO models

<d> ::= 0 to (# digital channels - 1) in NR1 format

<n> ::= 1 to (# of analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:BRATe command installs a screen measurement and starts the bit rate measurement. If the optional source parameter is specified, the currently specified source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**    `:MEASure:BRATe? [<source>]`

The :MEASure:BRATe? query measures all positive and negative pulse widths on the waveform, takes the minimum value found of either width type and inverts that minimum width to give a value in Hertz.

**Return Format**    `<value><NL>`

```
<value> ::= the bit rate value in Hertz
```

**See Also**

- ["Introduction to :MEASure Commands"](#) on page 476

- [":MEASure:SOURce"](#) on page 517
- [":MEASure:FREQuency"](#) on page 498
- [":MEASure:PERiod"](#) on page 506

## :MEASure:BWIDth

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:BWIDth [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMMEmory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:BWIDth command installs a burst width measurement on screen. If the optional source parameter is not specified, the current measurement source is used.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:BWIDth? [<source>]
```

The :MEASure:BWIDth? query measures and returns the width of the burst on the screen.

The burst width is calculated as follows:

burst width = (last edge on screen - first edge on screen)

### Return Format

```
<value><NL>
<value> ::= burst width in seconds in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:CLEar

**N** (see [page 1334](#))

**Command Syntax** :MEASure:CLEar

This command clears all selected measurements and markers from the screen.

**See Also** • ["Introduction to :MEASure Commands"](#) on page 476

## :MEASure:COUNter

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:MEASure:COUNter [&lt;source&gt;]</code>
	<code>&lt;source&gt; ::= {&lt;digital channels&gt;   CHANnel&lt;n&gt;   EXTERNAL}</code>
	<code>&lt;digital channels&gt; ::= DIGItal&lt;d&gt; for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :MEASure:COUNter command installs a screen measurement and starts a counter measurement. If the optional source parameter is specified, the current source is modified. Any channel except Math or Reference Waveforms may be selected for the source.
	The counter measurement counts trigger level crossings within a certain amount of time (gate time) and displays the results in Hz.
	The gate time is the horizontal range of the oscilloscope but is limited to $\geq 0.1$ s and $\leq 10$ s. Unlike other measurements, the Zoom horizontal timebase window does not gate the Counter measurement.
	The Counter measurement can measure frequencies up to the bandwidth of the oscilloscope. The minimum frequency supported is $2.0 / \text{gateTime}$ .
	Only one counter measurement may be displayed at a time.

### NOTE

This command is not available if the source is MATH.

### Query Syntax

`:MEASure:COUNter? [<source>]`

The :MEASure:COUNter? query measures and outputs the counter frequency of the specified source.

### NOTE

The :MEASure:COUNter? query times out if the counter measurement is installed on the front panel. Use :MEASure:CLEar to remove the front-panel measurement before executing the :MEASure:COUNter? query.

### Return Format

```
<source><NL>
<source> ::= count in Hertz in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517
- "[":MEASure:FREQuency](#)" on page 498

- [":MEASure:CLEar"](#) on page 482

## :MEASure:DEFine

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:DEFine <meas_spec>[,<source>]
<meas_spec> ::= {DEDelay | THresholds}
```

The :MEASure:DEFine command sets up the definition for measurements by specifying the delta time or threshold values. Changing these values may affect the results of other measure commands. The table below identifies which measurement results that can be affected by redefining the DEDelay specification or the THresholds values. For example, changing the THresholds definition from the default 10%, 50%, and 90% values may change the returned measurement result.

MEASure Command	DEDelay	THresholds
DUTYcycle		x
DEDelay	x	x
FALLtime		x
FREQency		x
NWIDth		x
OVERshoot		x
PERiod		x
PHASE		x
PRESHoot		x
PWIDth		x
RISetime		x
VAVerage		x
VRMS		x

### :MEASure:DEFine DEDelay Command Syntax

```
:MEASure:DEFine DELay,<delay spec>[,<source>]
<delay spec> ::= <edge_spec1>,<edge_spec2>
<edge_spec1> ::= [<slope>]<occurrence>
<edge_spec2> ::= [<slope>]<occurrence>
<slope> ::= {+ | -}
<occurrence> ::= integer
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | FFT | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
```

```
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

This command defines the behavior of the :MEASure:DELay? query by specifying the start and stop edge to be used. <edge\_spec1> specifies the slope and edge number on source1. <edge\_spec2> specifies the slope and edge number on source2. The measurement is taken as:

$$\text{delay} = t(\text{edge\_spec2}) - t(\text{edge\_spec1})$$

### NOTE

The :MEASure:DELay command and the front-panel delay measurement use an auto-edge selection method to determine the actual edge used for the measurement. The :MEASure:DEFine command has no effect on these delay measurements. The edges specified by the :MEASure:DEFine command only define the edges used by the :MEASure:DELay? query.

### :MEASure:DEFine THresholds Command Syntax

```
:MEASure:DEFine THresholds,<threshold spec>[,<source>]
<threshold spec> ::= {STANDARD
                      | {<threshold mode>,<upper>,<middle>,<lower>}}
<threshold mode> ::= {PERCENT | ABSOLUTE}
for <threshold mode> = PERCENT:
```

<upper>, <middle>, <lower> ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.

```
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | FFT | WMMemory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

for <threshold mode> = ABSOLUTE:

<upper>, <middle>, <lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.

- STANDARD threshold specification sets the lower, middle, and upper measurement thresholds to 10%, 50%, and 90% values between Vbase and Vtop.
- Threshold mode PERCENT sets the measurement thresholds to any user-defined percentages between 5% and 95% of values between Vbase and Vtop.
- Threshold mode ABSOLUTE sets the measurement thresholds to absolute values. ABSOLUTE thresholds are dependent on channel scaling (:CHANnel<n>:RANGE or "**:CHANnel<n>:SCALE**" on page 297:CHANnel<n>:SCALE), probe attenuation (:CHANnel<n>:PROBE), and probe units (:CHANnel<n>:UNITS). Always set these values first before setting ABSOLUTE thresholds.

<b>Query Syntax</b>	<code>:MEASure:DEFine? &lt;meas_spec&gt;[,&lt;source&gt;]</code> <code>&lt;meas_spec&gt; ::= {DEDelay   THresholds}</code>
	The :MEASURE:DEFINE? query returns the current edge specification for the delay measurements setup or the current specification for the thresholds setup.
<b>Return Format</b>	for <meas_spec> = DELay:  <code>{ &lt;edge_spec1&gt;   &lt;edge_spec2&gt;   &lt;edge_spec1&gt;,&lt;edge_spec2&gt;} &lt;NL&gt;</code>  for <meas_spec> = THresholds and <threshold mode> = PERCent:  <code>THR,PERC,&lt;upper&gt;,&lt;middle&gt;,&lt;lower&gt;&lt;NL&gt;</code> <code>&lt;upper&gt;, &lt;middle&gt;, &lt;lower&gt; ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.</code>  for <meas_spec> = THresholds and <threshold mode> = ABSolute:  <code>THR,ABS,&lt;upper&gt;,&lt;middle&gt;,&lt;lower&gt;&lt;NL&gt;</code> <code>&lt;upper&gt;, &lt;middle&gt;, &lt;lower&gt; ::= A number specifying the upper, middle, and lower threshold voltages in NR3 format.</code>  for <threshold spec> = STAndard:  <code>THR,PERC,+90.0,+50.0,+10.0</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :MEASure Commands</a>" on page 476</li> <li>· "<a href="#">":MEASure:DEDelay</a>" on page 488</li> <li>· "<a href="#">":MEASure:SOURce</a>" on page 517</li> <li>· "<a href="#">":CHANnel&lt;n&gt;:RANGE</a>" on page 296</li> <li>· "<a href="#">":CHANnel&lt;n&gt;:SCALe</a>" on page 297</li> <li>· "<a href="#">":CHANnel&lt;n&gt;:PROBe</a>" on page 289</li> <li>· "<a href="#">":CHANnel&lt;n&gt;:UNITS</a>" on page 298</li> </ul>

## :MEASure:DELay

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:DELay [<source1> [, <source2>]`

```
<source1>, <source2> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | WMEMory<r>}

<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:DELay command places the instrument in the continuous measurement mode and starts a delay measurement.

The measurement is taken as:

$$\text{delay} = t(\text{edge spec 2}) - t(\text{edge spec 1})$$

where the <edge spec> definitions are set by the :MEASure:DEFine command

### NOTE

The :MEASure:DELay command and the front-panel delay measurement differ from the :MEASure:DELay? query.

The delay command or front-panel measurement run the delay measurement in auto-edge select mode. In this mode, you can select the edge polarity, but the instrument will select the edges that will make the best possible delay measurement. The source1 edge chosen will be the edge that meets the polarity specified and is closest to the trigger reference point. The source2 edge selected will be that edge of the specified polarity that gives the first of the following criteria:

- The smallest positive delay value that is less than source1 period.
- The smallest negative delay that is less than source1 period.
- The smallest absolute value of delay.

The :MEASure:DELay? query will make the measurement using the edges specified by the :MEASure:DEFine command.

**Query Syntax**    `:MEASure:DELay? [<source1> [, <source2>]`

The :MEASure:DELay? query measures and returns the delay between source1 and source2. The delay measurement is made from the user-defined slope and edge count of the signal connected to source1, to the defined slope and edge count of the signal connected to source2. Delay measurement slope and edge parameters are selected using the :MEASure:DEFine command.

Also in the :MEASure:DEFine command, you can set upper, middle, and lower threshold values. *It is the middle threshold value that is used when performing the delay query.* The standard upper, middle, and lower measurement thresholds are

90%, 50%, and 10% values between Vbase and Vtop. If you want to move the delay measurement point nearer to Vtop or Vbase, you must change the threshold values with the :MEASure:DEFine THresholds command.

<b>Return Format</b>	<value><NL> <value> ::= floating-point number delay time in seconds in NR3 format
<b>See Also</b>	<ul style="list-style-type: none"><li>· "<a href="#">Introduction to :MEASure Commands</a>" on page 476</li><li>· "<a href="#">:MEASure:DEFine</a>" on page 485</li><li>· "<a href="#">:MEASure:PHASE</a>" on page 507</li></ul>

## :MEASure:DUAL:CHARge

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax** `:MEASure:DUAL:CHARge [<interval>[,<source1>][,<source2>]`

```
<interval> ::= {CYCLE | DISPLAY}
<source1>,<source2> ::= CHANnel<n> with N2820A probe connected
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:DUAL:CHARge command installs a charge measurement on screen. Charge measurements show the area between the waveform and the ground level.

The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax** `:MEASure:DUAL:CHARge? [<interval>[,<source1>][,<source2>]`

The :MEASure:DUAL:CHARge? query measures and returns the charge measurement value.

**Return Format** `<value><NL>`

```
<value> ::= the charge value in Amp-hours in NR3 format
```

**See Also**

- "[:MEASure:DUAL:VAMplitude](#)" on page 491
- "[:MEASure:DUAL:VAverage](#)" on page 492
- "[:MEASure:DUAL:VBASe](#)" on page 493
- "[:MEASure:DUAL:VPP](#)" on page 494
- "[:MEASure:DUAL:VRMS](#)" on page 495
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:DUAL:VAMPlitude

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax** `:MEASure:DUAL:VAMPlitude [<source1> [, <source2>]]`

`<source1>, <source2> ::= CHANnel<n> with N2820A probe connected`  
`<n> ::= 1 to (# analog channels) in NR1 format`

The :MEASure:DUAL:VAMPlitude command installs a screen measurement and starts a vertical amplitude measurement.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax** `:MEASure:DUAL:VAMPlitude? [<source1> [, <source2>]]`

The :MEASure:DUAL:VAMPlitude? query measures and returns the vertical amplitude of the waveform. To determine the amplitude, the instrument measures Vtop and Vbase, then calculates the amplitude as follows:

$$\text{vertical amplitude} = \text{Vtop} - \text{Vbase}$$

**Return Format** `<value><NL>`

`<value> ::= the amplitude of the selected waveform in NR3 format`

- See Also**
- "[:MEASure:DUAL:CHARge](#)" on page 490
  - "[:MEASure:DUAL:VAverage](#)" on page 492
  - "[:MEASure:DUAL:VBASe](#)" on page 493
  - "[:MEASure:DUAL:VPP](#)" on page 494
  - "[:MEASure:DUAL:VRMS](#)" on page 495
  - "[Introduction to :MEASure Commands](#)" on page 476
  - "[:MEASure:SOURce](#)" on page 517
  - "[:MEASure:VTOP](#)" on page 538

## :MEASure:DUAL:VAverage

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax** `:MEASure:DUAL:VAverage [<interval> [, <source1>] [, <source2>]`

```
<interval> ::= {CYCLE | DISPLAY}
<source1>, <source2> ::= CHANnel<n> with N2820A probe connected
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:DUAL:VAverage command installs a screen measurement and starts an average value measurement.

The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax** `:MEASure:DUAL:VAverage? [<interval>] [, <source1>] [, <source2>]`

The :MEASure:DUAL:VAverage? query returns the average value measurement.

**Return Format** `<value><NL>`  
`<value> ::= calculated average value in NR3 format`

**See Also**

- [":MEASure:DUAL:CHARge"](#) on page 490
- [":MEASure:DUAL:VAMPLitude"](#) on page 491
- [":MEASure:DUAL:VBASe"](#) on page 493
- [":MEASure:DUAL:VPP"](#) on page 494
- [":MEASure:DUAL:VRMS"](#) on page 495
- ["Introduction to :MEASure Commands"](#) on page 476
- [":MEASure:SOURce"](#) on page 517

## :MEASure:DUAL:VBASe

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax**

```
:MEASure:DUAL:VBASe [<source1>] [,<source2>]
<source1>, <source2> ::= CHANnel<n> with N2820A probe connected
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:DUAL:VBASe command installs a screen measurement and starts a waveform base value measurement.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax**

```
:MEASure:DUAL:VBASe? [<source1>] [,<source2>]
```

The :MEASure:DUAL:VBASe? query returns the vertical value at the base of the waveform. The base value of a pulse is normally not the same as the minimum value.

**Return Format**

```
<base_voltage><NL>
<base_voltage> ::= value at the base of the selected waveform in
NR3 format
```

**See Also**

- "[:MEASure:DUAL:CHARge](#)" on page 490
- "[:MEASure:DUAL:VAMPLitude](#)" on page 491
- "[:MEASure:DUAL:VAverage](#)" on page 492
- "[:MEASure:DUAL:VPP](#)" on page 494
- "[:MEASure:DUAL:VRMS](#)" on page 495
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:VTOP](#)" on page 538
- "[:MEASure:VMIN](#)" on page 533

## :MEASure:DUAL:VPP

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax**

```
:MEASure:DUAL:VPP [<source1>] [,<source2>]
<source1>, <source2> ::= CHANnel<n> with N2820A probe connected
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:DUAL:VPP command installs a screen measurement and starts a vertical peak-to-peak measurement.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax**

```
:MEASure:DUAL:VPP? [<source1>] [,<source2>]
```

The :MEASure:DUAL:VPP? query measures the maximum and minimum vertical value for the selected source, then calculates the vertical peak-to-peak value and returns that value. The peak-to-peak value (Vpp) is calculated with the following formula:

$$V_{pp} = V_{max} - V_{min}$$

Vmax and Vmin are the vertical maximum and minimum values present on the selected source.

**Return Format**

```
<value><NL>
<value> ::= vertical peak to peak value in NR3 format
```

**See Also**

- [":MEASure:DUAL:CHARge"](#) on page 490
- [":MEASure:DUAL:VAMPLitude"](#) on page 491
- [":MEASure:DUAL:VAVerage"](#) on page 492
- [":MEASure:DUAL:VBASe"](#) on page 493
- [":MEASure:DUAL:VRMS"](#) on page 495
- ["Introduction to :MEASure Commands"](#) on page 476
- [":MEASure:SOURce"](#) on page 517
- [":MEASure:VMAX"](#) on page 532
- [":MEASure:VMIN"](#) on page 533

## :MEASure:DUAL:VRMS

**N** (see [page 1334](#))

**Overview** This measurement is available with the N2820A high sensitivity current probe when both the Primary and Secondary probe cables are used. This measurement joins the Zoom In waveform data below the probe's clamp level with Zoom Out waveform data above the probe's clamp level to create the waveform on which the measurement is made.

**Command Syntax**

```
:MEASure:DUAL:VRMS [<interval>] [,<type>] [,<source1>] [,<source2>]
<interval> ::= {CYCLE | DISPLAY}
<type> ::= {AC | DC}
<source1>,<source2> ::= CHANnel<n> with N2820A probe connected
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:DUAL:VRMS command installs a screen measurement and starts an RMS value measurement.

The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

The <type> option lets you choose between a DC RMS measurement and an AC RMS measurement. If <type> is not specified, DC is implied.

If the optional source parameter(s) are specified, the currently specified source(s) are modified.

**Query Syntax**

```
:MEASure:DUAL:VRMS? [<interval>] [,<type>] [,<source1>] [,<source2>]
```

The :MEASure:DUAL:VRMS? query measures and outputs the RMS value measurement.

**Return Format**

```
<value><NL>
<value> ::= calculated dc RMS value in NR3 format
```

**See Also**

- "[:MEASure:DUAL:CHARge](#)" on page 490
- "[:MEASure:DUAL:VAMPLitude](#)" on page 491
- "[:MEASure:DUAL:VAVerage](#)" on page 492
- "[:MEASure:DUAL:VBASe](#)" on page 493
- "[:MEASure:DUAL:VPP](#)" on page 494
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:DUTYcycle

 (see [page 1334](#))

**Command Syntax**

```
:MEASure:DUTYcycle [<source>]
<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMemory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:DUTYcycle command installs a screen measurement and starts a positive duty cycle measurement on the current :MEASure:SOURce. If the optional source parameter is specified, the current source is modified.

### NOTE

The signal must be displayed to make the measurement. This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**

```
:MEASure:DUTYcycle? [<source>]
```

The :MEASure:DUTYcycle? query measures and outputs the positive duty cycle of the signal specified by the :MEASure:SOURce command. The value returned for the duty cycle is the ratio of the positive pulse width to the period. The positive pulse width and the period of the specified signal are measured, then the duty cycle is calculated with the following formula:

$$\text{+duty cycle} = (\text{+pulse width}/\text{period}) * 100$$

**Return Format**

```
<value><NL>
<value> ::= ratio of positive pulse width to period in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:PERiod](#)" on page 506
- "[:MEASure:PWIDth](#)" on page 510
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:NDUTy](#)" on page 499

**Example Code**

- "[Example Code](#)" on page 518

## :MEASure:FALLtime

**C** (see [page 1334](#))

### Command Syntax

```
:MEASure:FALLtime [<source>]
<source> ::= {CHANnel<n> | FUNCTION<m> | MATH<m> | WMMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:FALLtime command installs a screen measurement and starts a fall-time measurement. For highest measurement accuracy, set the sweep speed as fast as possible, while leaving the falling edge of the waveform on the display. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:FALLtime? [<source>]
```

The :MEASure:FALLtime? query measures and outputs the fall time of the displayed falling (negative-going) edge closest to the trigger reference. The fall time is determined by measuring the time at the upper threshold of the falling edge, then measuring the time at the lower threshold of the falling edge, and calculating the fall time with the following formula:

$$\text{fall time} = \text{time at lower threshold} - \text{time at upper threshold}$$

### Return Format

```
<value><NL>
<value> ::= time in seconds between the lower threshold and upper
           threshold in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:RISetime](#)" on page 514
- "[":MEASure:SOURce](#)" on page 517

## :MEASure:FREQuency



(see [page 1334](#))

### Command Syntax

```
:MEASure:FREQuency [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMory<r>}

<digital channels> ::= DIGItal<d> for the MSO models

<d> ::= 0 to (# digital channels - 1) in NR1 format

<n> ::= 1 to (# of analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:FREQuency command installs a screen measurement and starts a frequency measurement. If the optional source parameter is specified, the current source is modified.

IF the edge on the screen closest to the trigger reference is rising:

THEN frequency = 1/(time at trailing rising edge - time at leading rising edge)

ELSE frequency = 1/(time at trailing falling edge - time at leading falling edge)

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:FREQuency? [<source>]
```

The :MEASure:FREQuency? query measures and outputs the frequency of the cycle on the screen closest to the trigger reference.

### Return Format

```
<source><NL>

<source> ::= frequency in Hertz in NR3 format
```

### See Also

- ["Introduction to :MEASure Commands" on page 476](#)
- [":MEASure:SOURce" on page 517](#)
- [":MEASure:PERiod" on page 506](#)

### Example Code

- [":Example Code" on page 518](#)

## :MEASure:NDUTy

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:NDUTy [<source>]
<source> ::= {<digital channels> | CHANnel<n> | FUNCtion<m> | MATH<m>
              | WMEMemory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:NDUTy command installs a screen measurement and starts a negative duty cycle measurement on the current :MEASure:SOURce. If the optional source parameter is specified, the current source is modified.

### NOTE

The signal must be displayed to make the measurement. This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:NDUTy? [<source>]
```

The :MEASure:NDUTy? query measures and outputs the negative duty cycle of the signal specified by the :MEASure:SOURce command. The value returned for the duty cycle is the ratio of the negative pulse width to the period. The negative pulse width and the period of the specified signal are measured, then the duty cycle is calculated with the following formula:

$$\text{-duty cycle} = (\text{-pulse width}/\text{period}) * 100$$

### Return Format

```
<value><NL>
<value> ::= ratio of negative pulse width to period in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:PERiod](#)" on page 506
- "[:MEASure:NWIDth](#)" on page 502
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:DUTYcycle](#)" on page 496

## :MEASure:NEDGes

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:NEDGes [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMORY<r>}

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:NEDGes command installs a falling edge count measurement on screen. If the optional source parameter is not specified, the current source is measured.

**NOTE**

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**    `:MEASure:NEDGes? [<source>]`

The :MEASure:NEDGes? query measures and returns the on-screen falling edge count.

**Return Format**    `<value><NL>`

```
<value> ::= the falling edge count in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:NPULses

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:NPULses [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMMEmory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:NPULses command installs a falling pulse count measurement on screen. If the optional source parameter is not specified, the current source is measured.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:NPULses? [<source>]
```

The :MEASure:NPULses? query measures and returns the on-screen falling pulse count.

### Return Format

```
<value><NL>
<value> ::= the falling pulse count in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:NWIDth

 (see [page 1334](#))

### Command Syntax

```
:MEASure:NWIDth [<source>]
<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:NWIDth command installs a screen measurement and starts a negative pulse width measurement. If the optional source parameter is not specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:NWIDth? [<source>]
```

The :MEASure:NWIDth? query measures and outputs the width of the negative pulse on the screen closest to the trigger reference using the midpoint between the upper and lower thresholds.

FOR the negative pulse closest to the trigger point:

width = (time at trailing rising edge - time at leading falling edge)

### Return Format

```
<value><NL>
<value> ::= negative pulse width in seconds in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:PWidth](#)" on page 510
- "[:MEASure:PERiod](#)" on page 506

## :MEASure:OVERshoot

 (see [page 1334](#))

### Command Syntax

```
:MEASure:OVERshoot [<source>]
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | WMMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:OVERshoot command installs a screen measurement and starts an overshoot measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:OVERshoot? [<source>]
```

The :MEASure:OVERshoot? query measures and returns the overshoot of the edge closest to the trigger reference, displayed on the screen. The method used to determine overshoot is to make three different vertical value measurements: Vtop, Vbase, and either Vmax or Vmin, depending on whether the edge is rising or falling.

For a rising edge:

$$\text{overshoot} = ((\text{Vmax}-\text{Vtop}) / (\text{Vtop}-\text{Vbase})) \times 100$$

For a falling edge:

$$\text{overshoot} = ((\text{Vbase}-\text{Vmin}) / (\text{Vtop}-\text{Vbase})) \times 100$$

Vtop and Vbase are taken from the normal histogram of all waveform vertical values. The extremum of Vmax or Vmin is taken from the waveform interval right after the chosen edge, halfway to the next edge. This more restricted definition is used instead of the normal one, because it is conceivable that a signal may have more preshoot than overshoot, and the normal extremum would then be dominated by the preshoot of the following edge.

### Return Format

```
<overshoot><NL>
<overshoot>::= the percent of the overshoot of the selected waveform in
NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:PREShoot](#)" on page 509
- "[:MEASure:SOURce](#)" on page 517

- [":MEASure:VMAX" on page 532](#)
- [":MEASure:VTOP" on page 538](#)
- [":MEASure:VBASe" on page 531](#)
- [":MEASure:VMIN" on page 533](#)

## :MEASure:PEDGes

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:PEDGes [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMMemory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PEDGes command installs a rising edge count measurement on screen. If the optional source parameter is not specified, the current source is measured.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:PEDGes? [<source>]
```

The :MEASure:NEDGes? query measures and returns the on-screen rising edge count.

### Return Format

```
<value><NL>
<value> ::= the rising edge count in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:PERiod

(see [page 1334](#))

**Command Syntax**

```
:MEASure:PERiod [<source>]
<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMemory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PERiod command installs a screen measurement and starts the period measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**

```
:MEASure:PERiod? [<source>]
```

The :MEASure:PERiod? query measures and outputs the period of the cycle closest to the trigger reference on the screen. The period is measured at the midpoint of the upper and lower thresholds.

IF the edge closest to the trigger reference on screen is rising:

THEN period = (time at trailing rising edge - time at leading rising edge)

ELSE period = (time at trailing falling edge - time at leading falling edge)

**Return Format**

```
<value><NL>
```

<value> ::= waveform period in seconds in NR3 format

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:NWIDth](#)" on page 502
- "[:MEASure:PVIDth](#)" on page 510
- "[:MEASure:FREQuency](#)" on page 498

**Example Code**

- "[Example Code](#)" on page 518

## :MEASure:PHASE

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:PHASE [<source1> [,<source2>]`

`<source1>, <source2> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMory<r>}`

`<n> ::= 1 to (# analog channels) in NR1 format`

`<m> ::= 1 to (# math functions) in NR1 format`

`<r> ::= 1 to (# ref waveforms) in NR1 format`

The :MEASure:PHASE command places the instrument in the continuous measurement mode and starts a phase measurement.

**Query Syntax**    `:MEASure:PHASE? [<source1> [,<source2>]`

The :MEASure:PHASE? query measures and returns the phase between the specified sources.

A phase measurement is a combination of the period and delay measurements. First, the period is measured on source1. Then the delay is measured between source1 and source2. The edges used for delay are the source1 rising edge used for the period measurement closest to the horizontal reference and the rising edge on source 2. See :MEASure:DELay for more detail on selecting the 2nd edge.

The phase is calculated as follows:

$$\text{phase} = (\text{delay} / \text{period of input 1}) \times 360$$

**Return Format**    `<value><NL>`

`<value> ::= the phase angle value in degrees in NR3 format`

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:DELay](#)" on page 488
- "[:MEASure:PERiod](#)" on page 506
- "[:MEASure:SOURce](#)" on page 517

## :MEASure:PPULses

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:PPULses [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PPULses command installs a rising pulse count measurement on screen. If the optional source parameter is not specified, the current source is measured.

**NOTE**

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**    `:MEASure:PPULses? [<source>]`

The :MEASure:PPULses? query measures and returns the on-screen rising pulse count.

**Return Format**    `<value><NL>`

```
<value> ::= the rising pulse count in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517

## :MEASure:PREShoot



(see [page 1334](#))

**Command Syntax**

```
:MEASure:PREShoot [<source>]
<source> ::= {CHANnel<n> | FUNCTION<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PREShoot command installs a screen measurement and starts a preshoot measurement. If the optional source parameter is specified, the current source is modified.

**Query Syntax**

```
:MEASure:PREShoot? [<source>]
```

The :MEASure:PREShoot? query measures and returns the preshoot of the edge closest to the trigger, displayed on the screen. The method used to determine preshoot is to make three different vertical value measurements: Vtop, Vbase, and either Vmin or Vmax, depending on whether the edge is rising or falling.

For a rising edge:

$$\text{preshoot} = ((V_{\text{min}} - V_{\text{base}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

For a falling edge:

$$\text{preshoot} = ((V_{\text{max}} - V_{\text{top}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

Vtop and Vbase are taken from the normal histogram of all waveform vertical values. The extremum of Vmax or Vmin is taken from the waveform interval right before the chosen edge, halfway back to the previous edge. This more restricted definition is used instead of the normal one, because it is likely that a signal may have more overshoot than preshoot, and the normal extremum would then be dominated by the overshoot of the preceding edge.

**Return Format**

```
<value><NL>
<value> ::= the percent of preshoot of the selected waveform
           in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:VMIN](#)" on page 533
- "[:MEASure:VMAX](#)" on page 532
- "[:MEASure:VTOP](#)" on page 538
- "[:MEASure:VBASE](#)" on page 531

## :MEASure:PWIDth

 (see [page 1334](#))

**Command Syntax**

```
:MEASure:PWIDth [<source>]
<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
              | WMEMemory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PWIDth command installs a screen measurement and starts the positive pulse width measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**

```
:MEASure:PWIDth? [<source>]
```

The :MEASure:PWIDth? query measures and outputs the width of the displayed positive pulse closest to the trigger reference. Pulse width is measured at the midpoint of the upper and lower thresholds.

IF the edge on the screen closest to the trigger is falling:

THEN width = (time at trailing falling edge - time at leading rising edge)

ELSE width = (time at leading falling edge - time at leading rising edge)

**Return Format**

```
<value><NL>
<value> ::= width of positive pulse in seconds in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:NWIDth](#)" on page 502
- "[:MEASure:PERiod](#)" on page 506

## :MEASure:RESults

**N** (see [page 1334](#))

### Query Syntax :MEASure:RESults?

The :MEASure:RESults? query returns the results of the continuously displayed measurements. The response to the MEASure:RESults? query is a list of comma-separated values.

If more than one measurement is running continuously, the :MEASure:RESults return values are duplicated for each continuous measurement from the first to last (top to bottom) result displayed. Each result returned is separated from the previous result by a comma. There is a maximum of 10 continuous measurements that can be continuously displayed at a time.

When no quick measurements are installed, the :MEASure:RESults? query returns nothing (empty string). When the count for any of the measurements is 0, the value of infinity (9.9E+37) is returned for the min, max, mean, and standard deviation.

### Return Format <result\_list><NL>

<result\_list> ::= comma-separated list of measurement results

The following shows the order of values received for a single measurement if :MEASure:STATistics is set to ON.

Measureme nt label	current	min	max	mean	std dev	count
--------------------	---------	-----	-----	------	---------	-------

Measurement label, current, min, max, mean, std dev, and count are only returned if :MEASure:STATistics is ON.

If :MEASure:STATistics is set to CURRent, MIN, MAX, MEAN, STDDev, or COUNT only that particular statistic value is returned for each measurement that is on.

- See Also**
- ["Introduction to :MEASure Commands" on page 476](#)
  - [":MEASure:STATistics" on page 519](#)

### Example Code

```
' This program shows the InfiniiVision oscilloscopes' measurement
' statistics commands.
' -----
'
```

#### Option Explicit

```
Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

```

Sub Main()

    On Error GoTo VisaComError

    ' Create the VISA COM I/O resource.
    Set myMgr = New VisaComLib.ResourceManager
    Set myScope = New VisaComLib.FormattedIO488
    Set myScope.IO = myMgr.Open("TCPIP0::130.29.70.228::inst0::INSTR")

    ' Initialize.
    myScope.IO.Clear      ' Clear the interface.
    myScope.WriteString "*RST"      ' Reset to the defaults.
    myScope.WriteString "*CLS"      ' Clear the status data structures.
    myScope.WriteString ":AUToscale"

    ' Install some measurements.
    myScope.WriteString ":MEASure:SOURce CHANnel1"      ' Input source.

    Dim MeasurementArray(3) As String
    MeasurementArray(0) = "FREQuency"
    MeasurementArray(1) = "DUTYcycle"
    MeasurementArray(2) = "VAMPplitude"
    MeasurementArray(3) = "VPP"
    Dim Measurement As Variant

    For Each Measurement In MeasurementArray
        myScope.WriteString ":MEASure:" + Measurement
        myScope.WriteString ":MEASure:" + Measurement + "?"
        varQueryResult = myScope.ReadNumber      ' Read measurement value.
        Debug.Print Measurement + ":" + FormatNumber(varQueryResult, 4)
    Next

    myScope.WriteString ":MEASure:STATistics:RESET"      ' Reset stats.
    Sleep 5000      ' Wait for 5 seconds.

    ' Select the statistics results type.
    Dim ResultsTypeArray(6) As String
    ResultsTypeArray(0) = "CURRent"
    ResultsTypeArray(1) = "MINimum"
    ResultsTypeArray(2) = "MAXimum"
    ResultsTypeArray(3) = "MEAN"
    ResultsTypeArray(4) = "STDDev"
    ResultsTypeArray(5) = "COUNT"
    ResultsTypeArray(6) = "ON"      ' All results.
    Dim ResultType As Variant

    Dim ResultsList()

    Dim ValueColumnArray(6) As String
    ValueColumnArray(0) = "Meas_Lbl"
    ValueColumnArray(1) = "Current"
    ValueColumnArray(2) = "Min"
    ValueColumnArray(3) = "Max"
    ValueColumnArray(4) = "Mean"
    ValueColumnArray(5) = "Std_Dev"
    ValueColumnArray(6) = "Count"

```

```

Dim ValueColumn As Variant

For Each ResultType In ResultsTypeArray
    myScope.WriteString ":MEASure:STATistics " + ResultType

    ' Get the statistics results.
    Dim intCounter As Integer
    intCounter = 0
    myScope.WriteString ":MEASure:REsults?"
    ResultsList() = myScope.ReadList

    For Each Measurement In MeasurementArray

        If ResultType = "ON" Then      ' All statistics.

            For Each ValueColumn In ValueColumnArray
                If VarType(ResultsList(intCounter)) <> vbString Then
                    Debug.Print "Measure statistics result CH1, " +
                        Measurement + ", " + ValueColumn + ":" + -
                        FormatNumber(ResultsList(intCounter), 4)

                Else      ' Result is a string (e.g., measurement label).
                    Debug.Print "Measure statistics result CH1, " +
                        Measurement + ", " + ValueColumn + ":" + -
                        ResultsList(intCounter)

                End If

                intCounter = intCounter + 1

            Next

        Else      ' Specific statistic (e.g., Current, Max, Min, etc.).

            Debug.Print "Measure statistics result CH1, " +
                Measurement + ", " + ResultType + ":" + -
                FormatNumber(ResultsList(intCounter), 4)

            intCounter = intCounter + 1

        End If

    Next

    Exit Sub

VisaComError:
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

## :MEASure:RISetime

**C** (see [page 1334](#))

### Command Syntax

```
:MEASure: RISetime [<source>]
<source> ::= {CHANnel<n> | FUNCTION<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:RISetime command installs a screen measurement and starts a rise-time measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure: RISetime? [<source>]
```

The :MEASure:RISetime? query measures and outputs the rise time of the displayed rising (positive-going) edge closest to the trigger reference. For maximum measurement accuracy, set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The rise time is determined by measuring the time at the lower threshold of the rising edge and the time at the upper threshold of the rising edge, then calculating the rise time with the following formula:

$$\text{rise time} = \text{time at upper threshold} - \text{time at lower threshold}$$

### Return Format

```
<value><NL>
<value> ::= rise time in seconds in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517
- "[":MEASure:FALLtime](#)" on page 497

## :MEASure:SDEViation

**N** (see [page 1334](#))

### Command Syntax

```
:MEASure:SDEViation [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1-2 or 1-4 (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

### NOTE

This ":MEASure:VRMS DISPlay, AC" command is the preferred syntax for making standard deviation measurements.

The :MEASure:SDEViation command installs a screen measurement and starts std deviation measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:SDEViation? [<source>]
```

The :MEASure:SDEViation? query measures and outputs the std deviation of the selected waveform. The oscilloscope computes the std deviation on all displayed data points.

### Return Format

```
<value><NL>
<value> ::= calculated std deviation value in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:VRMS"](#) on page 536
- "[":MEASure:SOURce"](#) on page 517

**:MEASure:SHOW****N** (see [page 1334](#))**Command Syntax**    `:MEASure:SHOW <on_off>``<on_off> ::= {{0 | OFF} | {1 | ON}}`

The :MEASure:SHOW command enables markers for tracking measurements on the display.

**Query Syntax**    `:MEASure:SHOW?`

The :MEASure:SHOW? query returns the current state of the markers.

This can return OFF when :MARKer:MODE selects a mode other than MEASurement.

**Return Format**    `<on_off><NL>``<on_off> ::= {1 | 0}`**See Also**

- "Introduction to :MEASure Commands" on page 476
- ":MARKer:MODE" on page 443

## :MEASure:SOURce



(see [page 1334](#))

### Command Syntax

```
:MEASure:SOURce <source1>[,<source2>]
<source1>,<source2> ::= {<digital channels> | CHANnel<n> | FUNCtion
                           | MATH | WMEMory<r> | EXTernal}
<digital channels> ::= DIGItal<d> for the MSO models
<n> ::= 1 to (# of analog channels) in NR1 format
<r> ::= 1-2 in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :MEASure:SOURce command sets the default sources for measurements. The specified sources are used as the sources for the MEASure subsystem commands if the sources are not explicitly set with the command.

If a source is specified for any measurement, the current source is changed to this new value.

If :MARKer:MODE is set to OFF or MANual, setting :MEASure:SOURce to CHANnel<n>, FUNCtion, or MATH will also set :MARKer:X1Y1source to source1 and :MARKer:X2Y2source to source2.

EXTernal is only a valid source for the counter measurement (and <source1>).

### Query Syntax

```
:MEASure:SOURce?
```

The :MEASure:SOURce? query returns the current source selections. If source2 is not specified, the query returns "NONE" for source2. If all channels are off, the query returns "NONE,NONE". Source2 only applies to :MEASure:DELay and :MEASure:PHASe measurements.

### NOTE

MATH is an alias for FUNCtion. The query will return FUNC if the source is FUNCtion or MATH.

### Return Format

```
<source1>,<source2><NL>
<source1>,<source2> ::= {<digital channels> | CHAN<n> | FUNC | WMWM<r>
                           | EXT | NONE}
```

### See Also:

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MARKer:MODE](#)" on page 443
- "[":MARKer:X1Y1source](#)" on page 446
- "[":MARKer:X2Y2source](#)" on page 449
- "[":MEASure:DELay](#)" on page 488
- "[":MEASure:PHASe](#)" on page 507

**Example Code**

```

' MEASURE - The commands in the MEASURE subsystem are used to make
' measurements on displayed waveforms.
myScope.WriteString ":MEASURE:SOURCE CHANNEL1"      ' Source to measure.
myScope.WriteString ":MEASURE:FREQUENCY?"      ' Query for frequency.
varQueryResult = myScope.ReadNumber      ' Read frequency.
MsgBox "Frequency:" + vbCrLf _
+ FormatNumber(varQueryResult / 1000, 4) + " kHz"
myScope.WriteString ":MEASURE:DUTYCYCLE?"      ' Query for duty cycle.
varQueryResult = myScope.ReadNumber      ' Read duty cycle.
MsgBox "Duty cycle:" + vbCrLf _
+ FormatNumber(varQueryResult, 3) + "%"
myScope.WriteString ":MEASURE:RISETIME?"      ' Query for risetime.
varQueryResult = myScope.ReadNumber      ' Read risetime.
MsgBox "Risetime:" + vbCrLf _
+ FormatNumber(varQueryResult * 1000000, 4) + " us"
myScope.WriteString ":MEASURE:VPP?"      ' Query for Pk to Pk voltage.
varQueryResult = myScope.ReadNumber      ' Read VPP.
MsgBox "Peak to peak voltage:" + vbCrLf _
+ FormatNumber(varQueryResult, 4) + " V"
myScope.WriteString ":MEASURE:VMAX?"      ' Query for Vmax.
varQueryResult = myScope.ReadNumber      ' Read Vmax.
MsgBox "Maximum voltage:" + vbCrLf _
+ FormatNumber(varQueryResult, 4) + " V"

```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :MEASure:STATistics

**N** (see [page 1334](#))

**Command Syntax** :MEASure:STATistics <type>

```
<type> ::= {{ON | 1} | CURRent | MINimum | MAXimum | MEAN | STDDev  
| COUNT}
```

The :MEASure:STATistics command determines the type of information returned by the :MEASure:RESults? query. ON means all the statistics are on.

**Query Syntax** :MEASure:STATistics?

The :MEASure:STATistics? query returns the current statistics mode.

**Return Format** <type><NL>

```
<type> ::= {ON | CURR | MIN | MAX | MEAN | STDD | COUN}
```

- See Also**
- "[Introduction to :MEASure Commands](#)" on page 476
  - "[:MEASure:RESults](#)" on page 511
  - "[:MEASure:STATistics:DISPlay](#)" on page 520
  - "[:MEASure:STATistics:RESet](#)" on page 523
  - "[:MEASure:STATistics:INCReement](#)" on page 521

- Example Code**
- "[Example Code](#)" on page 511

## :MEASure:STATistics:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:STATistics:DISPlay {{0 | OFF} | {1 | ON}}`

The :MEASure:STATistics:DISPlay command disables or enables the display of the measurement statistics.

**Query Syntax**    `:MEASure:STATistics:DISPlay?`

The :MEASure:STATistics:DISPlay? query returns the state of the measurement statistics display.

**Return Format**    `{0 | 1}<NL>`

**See Also**

- "Introduction to :MEASure Commands" on page 476

- "":MEASure:RESults" on page 511
- "":MEASure:STATistics" on page 519
- "":MEASure:STATistics:MCOut" on page 522
- "":MEASure:STATistics:RESet" on page 523
- "":MEASure:STATistics:INCReement" on page 521
- "":MEASure:STATistics:RSDeviation" on page 524

## :MEASure:STATistics:INCRement

**N** (see [page 1334](#))

**Command Syntax** :MEASure:STATistics:INCRement

This command updates the statistics once (incrementing the count by one) using the current measurement values. It corresponds to the front panel **Increment Statistics** softkey in the Measurement Statistics Menu. This command lets you, for example, gather statistics over multiple pulses captured in a single acquisition. To do this, change the horizontal position and enter the command for each new pulse that is measured.

This command is only allowed when the oscilloscope is stopped and quick measurements are on.

The command is allowed in segmented acquisition mode even though the corresponding front panel softkey is not available.

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:STATistics](#)" on page 519
- "[:MEASure:STATistics:DISPLAY](#)" on page 520
- "[:MEASure:STATistics:RESET](#)" on page 523
- "[:MEASure:RESULTS](#)" on page 511

**:MEASure:STATistics:MCOUNT****N** (see [page 1334](#))

**Command Syntax**    `:MEASure:STATistics:MCOUNT <setting>`  
`<setting> ::= {INFinite | <count>}`  
`<count> ::= 2 to 2000 in NR1 format`

The :MEASure:STATistics:MCOUNT command specifies the maximum number of values used when calculating measurement statistics.

**Query Syntax**    `:MEASure:STATistics:MCOUNT?`

The :MEASure:STATistics:MCOUNT? query returns the current measurement statistics max count setting.

**Return Format**    `<setting><NL>`  
`<setting> ::= {INF | <count>}`  
`<count> ::= 2 to 2000`

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:RESults](#)" on page 511
- "[":MEASure:STATistics](#)" on page 519
- "[":MEASure:STATistics:DISPlay](#)" on page 520
- "[":MEASure:STATistics:RSDeviation](#)" on page 524
- "[":MEASure:STATistics:RESet](#)" on page 523
- "[":MEASure:STATistics:INCRelement](#)" on page 521

## :MEASure:STATistics:RESet

**N** (see [page 1334](#))

**Command Syntax** :MEASure:STATistics:RESet

This command resets the measurement statistics, zeroing the counts.

Note that the measurement (statistics) configuration is not deleted.

**See Also** · ["Introduction to :MEASure Commands"](#) on page 476

· [":MEASure:STATistics"](#) on page 519

· [":MEASure:STATistics:DISPLAY"](#) on page 520

· [":MEASure:RESULTS"](#) on page 511

· [":MEASure:STATistics:INCREMENT"](#) on page 521

**Example Code** · ["Example Code"](#) on page 511

## :MEASure:STATistics:RSDeviation

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:STATistics:RSDeviation {{0 | OFF} | {1 | ON}}`

The :MEASure:STATistics:RSDeviation command disables or enables relative standard deviations, that is, standard deviation/mean, in the measurement statistics.

**Query Syntax**    `:MEASure:STATistics:RSDeviation?`

The :MEASure:STATistics:RSDeviation? query returns the current relative standard deviation setting.

**Return Format**    `{0 | 1}<NL>`

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:RESults](#)" on page 511
- "[:MEASure:STATistics](#)" on page 519
- "[:MEASure:STATistics:DISPlay](#)" on page 520
- "[:MEASure:STATistics:MCOunt](#)" on page 522
- "[:MEASure:STATistics:RESET](#)" on page 523
- "[:MEASure:STATistics:INCRelement](#)" on page 521

## :MEASure:TEDGE

**N** (see [page 1334](#))

**Query Syntax**

```
:MEASure:TEDGE? <slope><occurrence>[,<source>]

<slope> ::= direction of the waveform. A rising slope is indicated by a
           space or plus sign (+). A falling edge is indicated by a
           minus sign (-).

<occurrence> ::= the transition to be reported. If the occurrence number
                  is one, the first crossing from the left screen edge is
                  reported. If the number is two, the second crossing is
                  reported, etc.

<source> ::= {<digital channels> | CHANnel<n> | FUNCtion<m> | MATH<m>
              | WMEMory<r>}

<digital channels> ::= DIGItal<d> for the MSO models

<n> ::= 1 to (# of analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format

<d> ::= 0 to (# digital channels - 1) in NR1 format
```

When the :MEASure:TEDGE query is sent, the displayed signal is searched for the specified transition. The time interval between the trigger event and this occurrence is returned as the response to the query. The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the midpoint threshold in the positive direction. Once this crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified vertical value, or if the waveform does not cross the specified vertical value for the specific number of times in the direction specified.

You can make delay and phase measurements using the MEASure:TEDGE command:

Delay = time at the nth rising or falling edge of the channel - time at the same edge of another channel

Phase = (delay between channels / period of channel) x 360

For an example of making a delay and phase measurement, see "[:MEASure:TEDGE Code](#)" on page 526.

If the optional source parameter is specified, the current source is modified.

**NOTE**

This query is not available if the source is FFT (Fast Fourier Transform).

---

**Return Format**    <value><NL>  
 <value> ::= time in seconds of the specified transition in NR3 format

```
:MEASURE:TEDGE
  ' Make a delay measurement between channel 1 and 2.
  Dim dblChan1Edge1 As Double
  Dim dblChan2Edge1 As Double
  Dim dblChan1Edge2 As Double
  Dim dblDelay As Double
  Dim dblPeriod As Double
  Dim dblPhase As Double

  ' Query time at 1st rising edge on ch1.
  myScope.WriteString ":MEASURE:TEDGE? +1, CHAN1"

  ' Read time at edge 1 on ch 1.
  dblChan1Edge1 = myScope.ReadNumber

  ' Query time at 1st rising edge on ch2.
  myScope.WriteString ":MEASURE:TEDGE? +1, CHAN2"

  ' Read time at edge 1 on ch 2.
  dblChan2Edge1 = myScope.ReadNumber

  ' Calculate delay time between ch1 and ch2.
  dblDelay = dblChan2Edge1 - dblChan1Edge1

  ' Write calculated delay time to screen.
  MsgBox "Delay = " + vbCrLf + CStr(dblDelay)

  ' Make a phase difference measurement between channel 1 and 2.
  ' Query time at 1st rising edge on ch1.
  myScope.WriteString ":MEASURE:TEDGE? +2, CHAN1"

  ' Read time at edge 2 on ch 1.
  dblChan1Edge2 = myScope.ReadNumber

  ' Calculate period of ch 1.
  dblPeriod = dblChan1Edge2 - dblChan1Edge1

  ' Calculate phase difference between ch1 and ch2.
  dblPhase = (dblDelay / dblPeriod) * 360
  MsgBox "Phase = " + vbCrLf + CStr(dblPhase)
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 476
  - [":MEASure:TVALUE"](#) on page 527
  - [":MEASure:VTIMe"](#) on page 537

## :MEASure:TVALue

**C** (see [page 1334](#))

**Query Syntax**    `:MEASure:TVALue? <value>, [<slope>]<occurrence>[,<source>]`

`<value>` ::= the vertical value that the waveform must cross. The value can be volts or a math function value such as dB, Vs, or V/s.

`<slope>` ::= direction of the waveform. A rising slope is indicated by a plus sign (+). A falling edge is indicated by a minus sign (-).

`<occurrence>` ::= the transition to be reported. If the occurrence number is one, the first crossing is reported. If the number is two, the second crossing is reported, etc.

`<source>` ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | WMMemory<r>}

`<n>` ::= 1 to (# analog channels) in NR1 format

`<m>` ::= 1 to (# math functions) in NR1 format

`<r>` ::= 1 to (# ref waveforms) in NR1 format

When the :MEASure:TVALue? query is sent, the displayed signal is searched for the specified value level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to the query.

The specified value can be negative or positive. To specify a negative value, use a minus sign (-). The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of the occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the specified value level in the positive direction. Once this value crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified value, or if the waveform does not cross the specified value for the specified number of times in the direction specified.

If the optional source parameter is specified, the current source is modified.

### NOTE

This query is not available if the source is FFT (Fast Fourier Transform).

**Return Format**    `<value><NL>`

<value> ::= time in seconds of the specified value crossing in  
NR3 format

See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:TEDGE](#)" on page 525
- "[":MEASure:VTIMe](#)" on page 537

## :MEASure:VAMPlitude

**C** (see [page 1334](#))

**Command Syntax**    `:MEASure:VAMPlitude [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMMEmory<r>}

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VAMPlitude command installs a screen measurement and starts a vertical amplitude measurement. If the optional source parameter is specified, the current source is modified.

**Query Syntax**    `:MEASure:VAMPlitude? [<source>]`

The :MEASure:VAMPlitude? query measures and returns the vertical amplitude of the waveform. To determine the amplitude, the instrument measures Vtop and Vbase, then calculates the amplitude as follows:

$$\text{vertical amplitude} = \text{Vtop} - \text{Vbase}$$

**Return Format**    `<value><NL>`

```
<value> ::= the amplitude of the selected waveform in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:VBASe](#)" on page 531
- "[:MEASure:VTOP](#)" on page 538
- "[:MEASure:VPP](#)" on page 534

## :MEASure:VAverage

 (see [page 1334](#))

- |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Command Syntax</b> | <pre>:MEASure:VAverage [&lt;interval&gt;] [,&lt;source&gt;] &lt;interval&gt; ::= {CYCLE   DISPLAY} &lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   FFT   WMEMORY&lt;r&gt;} &lt;n&gt; ::= 1-2 or 1-4 (# of analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</pre> <p>The :MEASure:VAverage command installs a screen measurement and starts an average value measurement.</p> <p>The &lt;interval&gt; option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If &lt;interval&gt; is not specified, DISPLAY is implied.</p> <p>If the optional source parameter is specified, the current source is modified.</p> |
| <b>Query Syntax</b>   | <pre>:MEASure:VAverage? [&lt;interval&gt;] [,&lt;source&gt;]</pre> <p>The :MEASure:VAverage? query returns the average value of an integral number of periods of the signal.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Return Format</b>  | <pre>&lt;value&gt;&lt;NL&gt; &lt;value&gt; ::= calculated average value in NR3 format</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>See Also</b>       | <ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :MEASure Commands</a>" on page 476</li> <li>· "<a href="#">":MEASure:SOURce</a>" on page 517</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## :MEASure:VBASE

**C** (see [page 1334](#))

### Command Syntax

```
:MEASure:VBASE [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VBASE command installs a screen measurement and starts a waveform base value measurement. If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

```
:MEASure:VBASE? [<source>]
```

The :MEASure:VBASE? query returns the vertical value at the base of the waveform. The base value of a pulse is normally not the same as the minimum value.

### Return Format

```
<base_voltage><NL>
<base_voltage> ::= value at the base of the selected waveform in
NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:VTOP](#)" on page 538
- "[:MEASure:VAMPLitude](#)" on page 529
- "[:MEASure:VMIN](#)" on page 533

## :MEASure:VMAX



(see [page 1334](#))

### Command Syntax

```
:MEASure:VMAX [<source>]

<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | FFT | WMEMory<r>}

<n> ::= 1-2 or 1-4 (# of analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VMAX command installs a screen measurement and starts a maximum vertical value measurement. If the optional source parameter is specified, the current source is modified.

### Query Syntax

```
:MEASure:VMAX? [<source>]
```

The :MEASure:VMAX? query measures and outputs the maximum vertical value present on the selected waveform.

### Return Format

```
<value><NL>

<value> ::= maximum vertical value of the selected waveform in
NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517
- "[":MEASure:VMIN](#)" on page 533
- "[":MEASure:VPP](#)" on page 534
- "[":MEASure:VTOP](#)" on page 538

## :MEASure:VMIN

 (see [page 1334](#))

### Command Syntax

```
:MEASure:VMIN [<source>]
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | FFT | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VMIN command installs a screen measurement and starts a minimum vertical value measurement. If the optional source parameter is specified, the current source is modified.

### Query Syntax

```
:MEASure:VMIN? [<source>]
```

The :MEASure:VMIN? query measures and outputs the minimum vertical value present on the selected waveform.

### Return Format

```
<value><NL>
<value> ::= minimum vertical value of the selected waveform in
NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517
- "[":MEASure:VBASe](#)" on page 531
- "[":MEASure:VMAX](#)" on page 532
- "[":MEASure:VPP](#)" on page 534

## :MEASure:VPP

 (see [page 1334](#))

### Command Syntax

```
:MEASure:VPP [<source>]
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VPP command installs a screen measurement and starts a vertical peak-to-peak measurement. If the optional source parameter is specified, the current source is modified.

### Query Syntax

```
:MEASure:VPP? [<source>]
```

The :MEASure:VPP? query measures the maximum and minimum vertical value for the selected source, then calculates the vertical peak-to-peak value and returns that value. The peak-to-peak value (Vpp) is calculated with the following formula:

$$V_{pp} = V_{max} - V_{min}$$

Vmax and Vmin are the vertical maximum and minimum values present on the selected source.

### Return Format

```
<value><NL>
<value> ::= vertical peak to peak value in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MEASure:SOURce](#)" on page 517
- "[:MEASure:VMAX](#)" on page 532
- "[:MEASure:VMIN](#)" on page 533
- "[:MEASure:VAMPLitude](#)" on page 529

## :MEASure:VRATio

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:VRATio [<interval>[,<source1>[,<source2>]]`

```
<interval> ::= {CYCLE | DISPLAY}
<source1,2> ::= {CHANnel<n> | FUNCTION<m> | MATH<m> | WMEMORY<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VRATio command installs a ratio measurement on screen. Ratio measurements show the ratio of the ACRMS value of source1 to that of source2, expressed in dB.

The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

### Query Syntax

`:MEASure:VRATio? [<interval>][<source1>][,<source2>]`

The :MEASure:VRATio? query measures and returns the ratio of AC RMS values of the specified sources expressed as dB.

### Return Format

```
<value><NL>
<value> ::= the ratio value in dB in NR3 format
```

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:VRMS](#)" on page 536
- "[":MEASure:SOURce](#)" on page 517

## :MEASure:VRMS

**C** (see [page 1334](#))

**Command Syntax**

```
:MEASure:VRMS [<interval>[,<type>][,<source>]
<interval> ::= {CYCLE | DISPLAY}
<type> ::= {AC | DC}
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMMemory<r>}
<n> ::= 1-2 or 1-4 (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VRMS command installs a screen measurement and starts an RMS value measurement.

The <interval> option lets you specify the measurement interval: either an integral number of cycles, or the full screen. If <interval> is not specified, DISPLAY is implied.

The <type> option lets you choose between a DC RMS measurement and an AC RMS measurement. If <type> is not specified, DC is implied.

If the optional source parameter is specified, the current source is modified.

### NOTE

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**

```
:MEASure:VRMS? [<interval>[,<type>][,<source>]
```

The :MEASure:VRMS? query measures and outputs the RMS measurement value.

**Return Format**

```
<value><NL>
<value> ::= calculated dc RMS value in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:SOURce](#)" on page 517

## :MEASure:VTIMe

**N** (see [page 1334](#))

**Query Syntax**

```
:MEASure:VTIMe? <vtimetime_argument>[,<source>]
<vtimetime_argument> ::= time from trigger in seconds
<source> ::= {<digital channels> | CHANnel<n> | FUNCtion<m> | MATH<m>
              | FFT | WMEMory<r>}
<digital channels> ::= DIGItal<d> for the MSO models
<d> ::= 0 to (# digital channels - 1) in NR1 format
<n> ::= 1 to (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VTIMe? query returns the vertical value at a specified horizontal value on the source specified (see also :MEASure:SOURce). The specified horizontal value must be on the screen; when it is a time value, it is referenced to the trigger event. If the optional source parameter is specified, the measurement source is modified.

### NOTE

When the source is an FFT (Fast Fourier Transform) waveform, the <vtimetime\_argument> is a frequency value instead of a time value.

**Return Format**

```
<value><NL>
<value> ::= vertical value at the specified horizontal location
           in NR3 format
```

**See Also**

- ["Introduction to :MEASure Commands"](#) on page 476
- [":MEASure:SOURce"](#) on page 517
- [":MEASure:TEDGE"](#) on page 525
- [":MEASure:TVALUE"](#) on page 527

## :MEASure:VTOP

 (see [page 1334](#))

**Command Syntax**    `:MEASure:VTOP [⟨source⟩]`

```

⟨source⟩ ::= {CHANnel⟨n⟩ | FUNCtion⟨m⟩ | MATH⟨m⟩}

⟨n⟩ ::= 1-2 or 1-4 (# of analog channels) in NR1 format

⟨m⟩ ::= 1 to (# math functions) in NR1 format

⟨r⟩ ::= 1 to (# ref waveforms) in NR1 format

```

The :MEASure:VTOP command installs a screen measurement and starts a waveform top value measurement.

**NOTE**

This query is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**    `:MEASure:VTOP? [⟨source⟩]`

The :MEASure:VTOP? query returns the vertical value at the top of the waveform. The top value of the pulse is normally not the same as the maximum value.

**Return Format**    `<value><NL>`

`<value>` ::= vertical value at the top of the waveform in NR3 format

- See Also**
- "[Introduction to :MEASure Commands](#)" on page 476
  - "[:MEASure:SOURce](#)" on page 517
  - "[:MEASure:VMAX](#)" on page 532
  - "[:MEASure:VAMPLitude](#)" on page 529
  - "[:MEASure:VBASE](#)" on page 531

## :MEASure:WINDOW

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:WINDOW <type>`

```
<type> ::= {MAIN | ZOOM | AUTO | GATE}
```

The :MEASure:WINDOW command lets you choose whether measurements are made in the Main window portion of the display, the Zoom window portion of the display (when the zoomed time base is displayed), or gated by the X1 and X2 cursors.

- MAIN – the measurement window is the Main window.
- ZOOM – the measurement window is the lower, Zoom window.
- AUTO – when the zoomed time base is displayed, the measurement is attempted in the lower, Zoom window; if it cannot be made there, or if the zoomed time base is not displayed, the Main window is used.
- GATE – the measurement window is between the X1 and X2 cursors. When the zoomed time base is displayed, the X1 and X2 cursors in the Zoom window portion of the display are used.

**Query Syntax**    `:MEASure:WINDOW?`

The :MEASure:WINDOW? query returns the current measurement window setting.

**Return Format**    `<type><NL>`

```
<type> ::= {MAIN | ZOOM | AUTO | GATE}
```

**See Also**    ["Introduction to :MEASure Commands" on page 476](#)

[":MEASure:SOURce" on page 517](#)

## :MEASure:XMAX

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:XMAX [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | WMEMory<r>}
<n> ::= 1-2 or 1-4 (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:XMAX command installs a screen measurement and starts an X-at-Max-Y measurement on the selected window. If the optional source parameter is specified, the current source is modified.

**NOTE**

:MEASure:XMAX is an alias for :MEASure:TMAX.

**Query Syntax**    `:MEASure:XMAX? [<source>]`

The :MEASure:XMAX? query measures and returns the horizontal axis value at which the maximum vertical value occurs. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

**Return Format**    `<value><NL>`

```
<value> ::= horizontal value of the maximum in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:XMIN"](#) on page 541
- "[":MEASure:TMAX"](#) on page 1264

## :MEASure:XMIN

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:XMIN [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | FFT | WMMemory<r>}
<n> ::= 1-2 or 1-4 (# of analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:XMIN command installs a screen measurement and starts an X-at-Min-Y measurement on the selected window. If the optional source parameter is specified, the current source is modified.

**NOTE**

:MEASure:XMIN is an alias for :MEASure:TMIN.

**Query Syntax**    `:MEASure:XMIN? [<source>]`

The :MEASure:XMIN? query measures and returns the horizontal axis value at which the minimum vertical value occurs. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

**Return Format**    `<value><NL>`

```
<value> ::= horizontal value of the minimum in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:XMAX"](#) on page 540
- "[":MEASure:TMIN"](#) on page 1265



## 23 :MEASure Power Commands

These :MEASure commands are available when the DSOX3PWR power measurements and analysis application is licensed and enabled.

**Table 104** :MEASure Power Commands Summary

Command	Query	Options and Query Returns
:MEASure:ANGLE [<source1> [, <source2>] (see <a href="#">page 547</a> )	:MEASure:ANGLE? [<source1> [, <source2>] (see <a href="#">page 547</a> )	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the power phase angle in degrees in NR3 format
:MEASure:APPARENT [<source1> [, <source2>] (see <a href="#">page 548</a> )	:MEASure:APPARENT? [<source1> [, <source2>] (see <a href="#">page 548</a> )	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the apparent power value in NR3 format
:MEASure:CPLoss [<source1> [, <source2>] (see <a href="#">page 549</a> )	:MEASure:CPLoss? [<source1> [, <source2>] (see <a href="#">page 549</a> )	<source1>, <source2> <source1> ::= {FUNCTION<m>   MATH<m>} <source2> ::= {CHANnel<n>} <m> ::= 1 to (# math functions) in NR1 format <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the switching loss per cycle watts value in NR3 format

**Table 104:**MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:CRESt [<source>] (see page 550)	:MEASure:CRESt? [<source>] (see page 550)	<source> ::= {CHANnel<n>   FUNCTion<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= the crest factor value in NR3 format
:MEASure:EFFECTivity (see page 551)	:MEASure:EFFECTivity? (see page 551)	<return_value> ::= percent value in NR3 format
:MEASure:ELOSSs [<source>] (see page 552)	:MEASure:ELOSSs? [<source>] (see page 552)	<source> ::= {CHANnel<n>   FUNCTion<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the energy loss value in NR3 format
:MEASure:FACTOr [<source1> [, <source2>] (see page 553)	:MEASure:FACTOr? [<source1> [, <source2>] (see page 553)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the power factor value in NR3 format
:MEASure:IPower (see page 554)	:MEASure:IPower? (see page 554)	<return_value> ::= the input power value in NR3 format
:MEASure:OFFTime [<source1> [, <source2>] (see page 555)	:MEASure:OFFTime? [<source1> [, <source2>] (see page 555)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the time in seconds in NR3 format
:MEASure:ONTime [<source1> [, <source2>] (see page 556)	:MEASure:ONTime? [<source1> [, <source2>] (see page 556)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the time in seconds in NR3 format
:MEASure:OPOWer (see page 557)	:MEASure:OPOWer? (see page 557)	<return_value> ::= the output power value in NR3 format

**Table 104:** MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PCURrent [<source?>] (see page 558)	:MEASure:PCURrent? [<source>] (see page 558)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the peak current value in NR3 format</p>
:MEASure:PLOSS [<source>] (see page 559)	:MEASure:PLOSS? [<source>] (see page 559)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the power loss value in NR3 format</p>
:MEASure:RDSon [<source1> [, <source2>] (see page 560)	:MEASure:RDSon? [<source1> [, <source2>] (see page 560)	<p>&lt;source1&gt;, &lt;source2&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>&lt;return_value&gt; ::= the VCE(sat) value in NR3 format</p>
:MEASure:REACTive [<source1> [, <source2>] (see page 561)	:MEASure:REACTive? [<source1> [, <source2>] (see page 561)	<p>&lt;source1&gt;, &lt;source2&gt; ::= {CHANnel&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;return_value&gt; ::= the reactive power value in NR3 format</p>

**Table 104:**MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:REAL [<source>] (see page 562)	:MEASure:REAL? [<source>] (see page 562)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= the real power value in NR3 format
:MEASure:RIPPLE [<source>] (see page 563)	:MEASure:RIPPLE? [<source>] (see page 563)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the output ripple value in NR3 format
:MEASure:TRESPonse [<source>] (see page 564)	:MEASure:TRESPonse? [<source>] (see page 564)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= time in seconds for the overshoot to settle back into the band in NR3 format
:MEASure:VCESat [<source>] (see page 565)	:MEASure:VCESat? [<source>] (see page 565)	<source> ::= {CHANnel<n>   FUNCTION<m>   MATH<m>   WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <return_value> ::= the VCE(sat) value in NR3 format

## :MEASure:ANGLE

**N** (see [page 1334](#))

**Command Syntax** :MEASure:ANGLE [<source1> [, <source2>]

```
<source1>, <source2> ::= {CHANnel<n>}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:ANGLE command installs a power phase angle measurement on screen.

The <source1> parameter is the channel probing voltage and the <source2> parameter is the channel probing current. These sources can also be specified by the :MEASure:SOURce command.

Phase angle is a measure of power quality. In the *power triangle* (the right triangle where  $\text{apparent\_power}^2 = \text{real\_power}^2 + \text{reactive\_power}^2$ ), phase angle is the angle between the apparent power and the real power, indicating the amount of reactive power. Small phase angles equate to less reactive power.

**Query Syntax** :MEASure:ANGLE? [<source1> [, <source2>]

The :MEASure:ANGLE query returns the measured power phase angle in degrees.

**Return Format** <return\_value><NL>

```
<return_value> ::= the power phase angle in degrees in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:QUALity:APPLy](#)" on page 649

## :MEASure:APPARENT

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:APPARENT [<source1> [,<source2>]`

`<source1>, <source2> ::= {CHANnel<n>}`

`<n> ::= 1 to (# analog channels) in NR1 format`

The :MEASure:APPARENT command installs an apparent power measurement on screen.

The <source1> parameter is the channel probing voltage and the <source2> parameter is the channel probing current. These sources can also be specified by the :MEASure:SOURce command.

Apparent power is a measure of power quality. It is the portion of AC line power flow due to stored energy which returns to the source in each cycle.

IRMS \* VRMS

**Query Syntax**    `:MEASure:APPARENT? [<source1> [,<source2>]]`

The :MEASure:APPARENT query returns the measured apparent power.

**Return Format**    `<return_value><NL>`

`<return_value> ::= the apparent power value in NR3 format`

**See Also**

- [":MEASure:SOURce" on page 517](#)
- [":POWER:QUALITY:APPLY" on page 649](#)

## :MEASure:CPLoss

**N** (see [page 1334](#))

**Command Syntax**

```
:MEASure:CPLoss [<source1> [,<source2>]
<source1> ::= {FUNCTION<m> | MATH<m>}
<source2> ::= {CHANnel<n>}
<m> ::= 1 to (# math functions) in NR1 format
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:CPLoss command installs a power loss per cycle measurement on screen.

The <source1> parameter is typically a math multiply waveform or other waveform that represents power (voltage \* current). This source can also be specified by the :MEASure:SOURce command.

Power loss per cycle is  $P_n = (V_{dsn} * I_{dn}) * (\text{Time range of zoom window}) * (\text{Counter measurement of the voltage of the switching signal})$ , where n is each sample.

This measurement operates when in zoom mode and the counter measurement is installed on the voltage of the switching signal.

**Query Syntax**

`:MEASure:CPLoss? [<source1> [,<source2>]`

The :MEASure:CPLoss query returns the switching loss per cycle in watts.

**Return Format**

`<return_value><NL>`

`<return_value> ::= the switching loss per cycle value in NR3 format`

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:SWITch:APPLY](#)" on page 672

## :MEASure:CRESt

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:CRESt [<source>]`

`<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m>}`

`<n> ::= 1 to (# analog channels) in NR1 format`

`<m> ::= 1 to (# math functions) in NR1 format`

The :MEASure:CRESt command installs a crest factor measurement on screen.

The <source> parameter is the channel probing current or voltage. This source can also be specified by the :MEASure:SOURce command.

Crest factor is a measure of power quality. It is the ratio between the instantaneous peak AC line current (or voltage) required by the load and the RMS current (or voltage). For example: Ipeak / IRMS or Vpeak / VRMS.

**Query Syntax**    `:MEASure:CRESt? [<source>]`

The :MEASure:CRESt query returns the measured crest factor.

**Return Format**    `<return_value><NL>`

`<return_value> ::= the crest factor value in NR3 format`

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:QUALity:APPLy](#)" on page 649

## :MEASure:EFFiciency

**N** (see [page 1334](#))

**Command Syntax** :MEASure:EFFiciency

The :MEASure:EFFiciency command installs an efficiency (output power / input power) measurement on screen.

Before sending this command or query, you must specify the channels probing the input voltage, input current, output voltage, and output current (using the :POWer:SIGNals:SOURce:VOLTage<i> and :POWer:SIGNals:SOURce:CURRent<i> commands) and you must perform the automated signals setup (using the :POWer:SIGNals:AUTosetup EFFiciency command).

**Query Syntax** :MEASure:EFFiciency?

The :MEASure:EFFiciency query returns the measured efficiency as a percent value.

**Return Format**

```
<return_value><NL>
<return_value> ::= percent value in NR3 format
```

**See Also**

- "[":POWer:SIGNals:SOURce:VOLTage<i>](#)" on page 669
- "[":POWer:SIGNals:SOURce:CURRent<i>](#)" on page 668
- "[":POWer:SIGNals:AUTosetup](#)" on page 651
- "[":POWer:EFFiciency:APPLy](#)" on page 620

## :MEASure:ELOSSs

**N** (see [page 1334](#))

**Command Syntax**

```
:MEASure:ELOSSs [<source>]
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:ELOSSs command installs an energy loss measurement on screen.

The <source> parameter is typically a math multiply waveform or other waveform that represents power (voltage \* current). This source can also be specified by the :MEASure:SOURce command.

Energy loss =  $\sum (V_{ds_n} * I_{d_n}) * \text{sample size}$ , where n is each sample.

**Query Syntax**

```
:MEASure:ELOSS? [<source>]
```

The :MEASure:ELOSS query returns the switching loss in joules.

**Return Format**

```
<return_value><NL>
<return_value> ::= the energy loss value in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:SWITch:APPLy](#)" on page 672

## :MEASure:FACTOr

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:FACTOr [<source1> [,<source2>]`  
`<source1>, <source2> ::= {CHANnel<n>}`  
`<n> ::= 1 to (# analog channels) in NR1 format`

The :MEASure:FACTOr command installs a power factor measurement on screen.

The <source1> parameter is the channel probing voltage and the <source2> parameter is the channel probing current. These sources can also be specified by the :MEASure:SOURce command.

Power factor is a measure of power quality. It is the ratio of the actual AC line power to the apparent power:

Real Power / Apparent Power

**Query Syntax**    `:MEASure:FACTOr? [<source1> [,<source2>]`

The :MEASure:FACTOr query returns the measured power factor.

**Return Format**    `<return_value><NL>`  
`<return_value> ::= the power factor value in NR3 format`

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:QUALity:APPLy](#)" on page 649

**:MEASure:IPower****N** (see [page 1334](#))**Command Syntax** `:MEASure:IPower`

The :MEASure:IPower command installs an input power measurement on screen.

Before sending this command or query, you must specify the channels probing the input voltage, input current, output voltage, and output current (using the :POWER:SIGNals:SOURce:VOLTage<i> and :POWER:SIGNals:SOURce:CURRent<i> commands) and you must perform the automated signals setup (using the :POWER:SIGNals:AUTosetup EFFiciency command).

**Query Syntax** `:MEASure:IPower?`

The :MEASure:IPower query returns the measured input power.

**Return Format** `<return_value><NL>`

`<return_value>` ::= the input power value in NR3 format

- See Also**
- "[":POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669
  - "[":POWER:SIGNals:SOURce:CURRent<i>](#)" on page 668
  - "[":POWER:SIGNals:AUTosetup](#)" on page 651
  - "[":POWER:EFFiciency:APPLy](#)" on page 620

## :MEASure:OFFTime

**N** (see [page 1334](#))

**Command Syntax**

```
:MEASure:OFFTime [<source1>[,<source2>]
<source1>, <source2> ::= {CHANnel<n>}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:OFFTime command installs an "off time" measurement on screen.

Turn off time measures the difference of time between when the input AC Voltage last falls to 10% of its maximum amplitude to the time when the output DC Voltage last falls to 10% of its maximum amplitude.

The <source1> parameter is the AC Voltage and the <source2> parameter is the DC Voltage. These sources can also be specified by the :MEASure:SOURce command.

**Query Syntax**

```
:MEASure:OFFTime? [<source1>[,<source2>]
```

The :MEASure:OFFTime query returns the measured turn off time.

**Return Format**

```
<return_value><NL>
<return_value> ::= the time in seconds in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:ONOFF:TEST](#)" on page 644
- "[:POWER:ONOFF:APPLy](#)" on page 641

## :MEASure:ONTime

**N** (see [page 1334](#))

**Command Syntax**

```
:MEASure:ONTIme [<source1> [,<source2>]
<source1>, <source2> ::= {CHANnel<n>}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:ONTIme command installs an "on time" measurement on screen.

Turn on time measures the difference of time between when the input AC Voltage first rises to 10% of its maximum amplitude to the time when the output DC Voltage rises to 90% of its maximum amplitude.

The <source1> parameter is the AC Voltage and the <source2> parameter is the DC Voltage. These sources can also be specified by the :MEASure:SOURce command.

**Query Syntax**

```
:MEASure:ONTIme? [<source1> [,<source2>]
```

The :MEASure:ONTIme query returns the measured turn off time.

**Return Format**

```
<return_value><NL>
<return_value> ::= the time in seconds in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:ONOFF:TEST](#)" on page 644
- "[:POWER:ONOFF:APPLy](#)" on page 641

## :MEASure:OPOWer

**N** (see [page 1334](#))

**Command Syntax** :MEASure:OPOWer

The :MEASure:OPOWer command installs an output power measurement on screen.

Before sending this command or query, you must specify the channels probing the input voltage, input current, output voltage, and output current (using the :POWER:SIGNals:SOURce:VOLTage<i> and :POWER:SIGNals:SOURce:CURRent<i> commands) and you must perform the automated signals setup (using the :POWER:SIGNals:AUTosetup EFFiciency command).

**Query Syntax** :MEASure:OPOWer?

The :MEASure:OPOWer query returns the measured output power.

**Return Format** <return\_value><NL>

<return\_value> ::= the output power value in NR3 format

- See Also**
- "[":POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669
  - "[":POWER:SIGNals:SOURce:CURRent<i>](#)" on page 668
  - "[":POWER:SIGNals:AUTosetup](#)" on page 651
  - "[":POWER:EFFiciency:APPLy](#)" on page 620

## :MEASure:PCURrent

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:PCURrent [<source>]`

```
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PCURrent command installs a peak current measurement on screen.

The <source> parameter is the channel probing the current. This source can also be specified by the :MEASure:SOURce command.

This command measures the peak current when the power supply first turned on.

**Query Syntax**    `:MEASure:PCURrent? [<source>]`

The :MEASure:PCURrent query returns the measured peak current.

**Return Format**    `<return_value><NL>`

```
<return_value> ::= the peak current value in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:INRush:APPLY](#)" on page 635

## :MEASure:PLOSS

**N** (see [page 1334](#))

**Command Syntax** :MEASure:PLOSS [<source>]

```
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:PLOSS command installs a power loss measurement on screen.

The <source> parameter is typically a math multiply waveform or other waveform that represents power (voltage \* current). This source can also be specified by the :MEASure:SOURce command.

Power loss is  $P_n = V_{ds,n} * I_{d,n}$ , where n is each sample.

**Query Syntax** :MEASure:PLOSS? [<source>]

The :MEASure:PLOSS query returns the switching loss in watts.

**Return Format** <return\_value><NL>

```
<return_value> ::= the power loss value in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:SWITch:APPLy](#)" on page 672

## :MEASure:RDSon

**N** (see [page 1334](#))

- Command Syntax**    `:MEASure:RDSon [<source1> [,<source2>]`
- ```
<source1>, <source2> ::= {CHANnel<n>| FUNCtion<m> | MATH<m>
| WMMemory<r>}
```
- `<n> ::= 1 to (# analog channels) in NR1 format`
- `<m> ::= 1 to (# math functions) in NR1 format`
- `<r> ::= 1 to (# ref waveforms) in NR1 format`
- The :MEASure:RDSon command installs a power Rds(on) measurement on screen.
- Rds(on) is the ON resistance between the drain and source of MOSFET. The Rds(on) characteristic is also published in the switching device data sheet.
- Query Syntax**    `:MEASure:RDSon? [<source1> [,<source2>]`
- The :MEASure:RDSon? query returns the measured Rds(on) value.
- Return Format**    `<return_value><NL>`
- ```
<return_value> ::= the VCE(sat) value in NR3 format
```
- See Also**
  - [":MEASure:VCESat"](#) on page 565

## :MEASure:REACTive

**N** (see [page 1334](#))

**Command Syntax** :MEASure:REACTive [<source1>] [,<source2>]

```
<source1>, <source2> ::= {CHANnel<n>}  
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:REACTive command installs a reactive power measurement on screen.

The <source1> parameter is the channel probing voltage and the <source2> parameter is the channel probing current. These sources can also be specified by the :MEASure:SOURce command.

Reactive power is a measure of power quality. It is the difference between apparent power and real power due to reactance. Using the *power triangle* (the right triangle where  $\text{apparent\_power}^2 = \text{real\_power}^2 + \text{reactive\_power}^2$ ):

$$\text{Reactive Power} = \sqrt{\text{Apparent Power}^2 - \text{Real Power}^2}$$

Reactive power is measured in VAR (Volts-Amps-Reactive).

**Query Syntax** :MEASure:REACTive? [<source1>] [,<source2>]

The :MEASure:REACTive query returns the measured reactive power.

**Return Format** <return\_value><NL>

```
<return_value> ::= the reactive power value in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:QUALITY:APPLy](#)" on page 649

## :MEASure:REAL

**N** (see [page 1334](#))

**Command Syntax**    `:MEASure:REAL [<source>]`

`<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m>}`

`<n> ::= 1 to (# analog channels) in NR1 format`

`<m> ::= 1 to (# math functions) in NR1 format`

The :MEASure:REAL command installs a real power measurement on screen.

The <source> parameter is typically a math multiply waveform or other waveform that represents power (voltage \* current). This source can also be specified by the :MEASure:SOURce command.

Real power is a measure of power quality. It is the portion of power flow that, averaged over a complete cycle of the AC waveform, results in net transfer of energy in one direction.

$$\text{Real Power} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} V_n I_n}$$

**Query Syntax**    `:MEASure:REAL? [<source>]`

The :MEASure:REAL query returns the measured real power.

**Return Format**    `<return_value><NL>`

`<return_value> ::= the real power value in NR3 format`

**See Also**

- [":MEASure:SOURce"](#) on page 517
- [":POWER:QUALITY:APPLY"](#) on page 649

## :MEASure:RIPPle

**N** (see [page 1334](#))

**Command Syntax** :MEASure:RIPPle [<source>]

```
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:RIPPle command installs an output ripple measurement on screen.

The <source> parameter is the channel probing the output voltage. This source can also be specified by the :MEASure:SOURce command.

Output ripple is: Vmax - Vmin.

**Query Syntax** :MEASure:RIPPle? [<source>]

The :MEASure:RIPPle query returns the measured output ripple.

**Return Format** <return\_value><NL>

```
<return_value> ::= the output ripple value in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:RIPPLE:APPLY](#)" on page 650

## :MEASure:TRESPonse

**N** (see [page 1334](#))

**Command Syntax**

```
:MEASure:TRESPonse [<source>]
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:TRESPonse command installs a transient response time measurement on screen.

The <source> parameter is the channel probing the output voltage. This source can also be specified by the :MEASure:SOURce command.

Transient response time =  $t_2 - t_1$ , where:

- $t_1$  = The first time a voltage waveform exits the settling band.
- $t_2$  = The last time it enters into the settling band.
- Settling band = +/-overshoot % of the steady state output voltage.

**Query Syntax**

```
:MEASure:TRESPonse? [<source>]
```

The :MEASure:TRESPonse query returns the measured transient response time.

**Return Format**

```
<return_value><NL>
<return_value> ::= time in seconds for the overshoot to settle back
                  into the band in NR3 format
```

**See Also**

- "[:MEASure:SOURce](#)" on page 517
- "[:POWER:TRANsient:APPLY](#)" on page 678

## :MEASure:VCESat

**N** (see [page 1334](#))

**Command Syntax** :MEASure:VCESat [<source>]

```
<source> ::= {CHANnel<n>| FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :MEASure:VCESat command installs a power Vce(sat) measurement on screen.

Vce(sat) is the saturation voltage between the collector and emitter of a BJT. The Vce(sat) characteristic is also published in the switching device data sheet.

**Query Syntax** :MEASure:VCESat? [<source>]

The :MEASure:VCESat? query returns the measured Vce(sat) value.

**Return Format** <return\_value><NL>

```
<return_value> ::= the VCE(sat) value in NR3 format
```

**See Also** • "[:MEASure:RDSon](#)" on page 560



## 24 :MTESt Commands

The MTESt subsystem commands and queries control the mask test features. See "[Introduction to :MTESt Commands](#)" on page 569.

**Table 105:**:MTESt Commands Summary

Command	Query	Options and Query Returns
:MTESt:ALL {{0   OFF}   {1   ON}} (see <a href="#">page 572</a> )	:MTESt:ALL? (see <a href="#">page 572</a> )	{0   1}
:MTESt:AMASK:CREAtE (see <a href="#">page 573</a> )	n/a	n/a
:MTESt:AMASK:SOURce <source> (see <a href="#">page 574</a> )	:MTESt:AMASK:SOURce? (see <a href="#">page 574</a> )	<source> ::= CHANnel<n> <n> ::= {1   2   3   4} for 4ch models <n> ::= {1   2} for 2ch models
:MTESt:AMASK:UNITS <units> (see <a href="#">page 575</a> )	:MTESt:AMASK:UNITS? (see <a href="#">page 575</a> )	<units> ::= {CURRent   DIVisions}
:MTESt:AMASK:XDELta <value> (see <a href="#">page 576</a> )	:MTESt:AMASK:XDELta? (see <a href="#">page 576</a> )	<value> ::= X delta value in NR3 format
:MTESt:AMASK:YDELta <value> (see <a href="#">page 577</a> )	:MTESt:AMASK:YDELta? (see <a href="#">page 577</a> )	<value> ::= Y delta value in NR3 format
n/a	:MTESt:COUNT:FWAVefor ms? [CHANnel<n>] (see <a href="#">page 578</a> )	<failed> ::= number of failed waveforms in NR1 format
:MTESt:COUNT:RESet (see <a href="#">page 579</a> )	n/a	n/a
n/a	:MTESt:COUNT:TIME? (see <a href="#">page 580</a> )	<time> ::= elapsed seconds in NR3 format
n/a	:MTESt:COUNT:WAVeform s? (see <a href="#">page 581</a> )	<count> ::= number of waveforms in NR1 format
:MTESt:DATA <mask> (see <a href="#">page 582</a> )	:MTESt:DATA? (see <a href="#">page 582</a> )	<mask> ::= data in IEEE 488.2 # format.

**Table 105:** MTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:MTEST:DELETE (see page 583)	n/a	n/a
:MTEST:ENABLE {{0   OFF}   {1   ON}} (see page 584)	:MTEST:ENABLE? (see page 584)	{0   1}
:MTEST:LOCK {{0   OFF}   {1   ON}} (see page 585)	:MTEST:LOCK? (see page 585)	{0   1}
:MTEST:RMODE <rmode> (see page 586)	:MTEST:RMODE? (see page 586)	<rmode> ::= {FORever   TIME   SIGMa   WAVEforms}
:MTEST:RMODE:FACTion:MEASure {{0   OFF}   {1   ON}} (see page 587)	:MTEST:RMODE:FACTion:MEASure? (see page 587)	{0   1}
:MTEST:RMODE:FACTion:PRINT {{0   OFF}   {1   ON}} (see page 588)	:MTEST:RMODE:FACTion:PRINT? (see page 588)	{0   1}
:MTEST:RMODE:FACTion:SAVE {{0   OFF}   {1   ON}} (see page 589)	:MTEST:RMODE:FACTion:SAVE? (see page 589)	{0   1}
:MTEST:RMODE:FACTion:STOP {{0   OFF}   {1   ON}} (see page 590)	:MTEST:RMODE:FACTion:STOP? (see page 590)	{0   1}
:MTEST:RMODE:SIGMa <level> (see page 591)	:MTEST:RMODE:SIGMa? (see page 591)	<level> ::= from 0.1 to 9.3 in NR3 format
:MTEST:RMODE:TIME <seconds> (see page 592)	:MTEST:RMODE:TIME? (see page 592)	<seconds> ::= from 1 to 86400 in NR3 format
:MTEST:RMODE:WAVEforms <count> (see page 593)	:MTEST:RMODE:WAVEforms? (see page 593)	<count> ::= number of waveforms in NR1 format
:MTEST:SCALe:BIND {{0   OFF}   {1   ON}} (see page 594)	:MTEST:SCALe:BIND? (see page 594)	{0   1}
:MTEST:SCALe:X1 <x1_value> (see page 595)	:MTEST:SCALe:X1? (see page 595)	<x1_value> ::= X1 value in NR3 format
:MTEST:SCALe:XDELta <xdelta_value> (see page 596)	:MTEST:SCALe:XDELta? (see page 596)	<xdelta_value> ::= X delta value in NR3 format

**Table 105:** MTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:MTEST:SCALe:Y1 <y1_value> (see page 597)	:MTEST:SCALe:Y1? (see page 597)	<y1_value> ::= Y1 value in NR3 format
:MTEST:SCALe:Y2 <y2_value> (see page 598)	:MTEST:SCALe:Y2? (see page 598)	<y2_value> ::= Y2 value in NR3 format
:MTEST:SOURce <source> (see page 599)	:MTEST:SOURce? (see page 599)	<source> ::= {CHANnel<n>   NONE} <n> ::= {1   2   3   4} for 4ch models <n> ::= {1   2} for 2ch models
n/a	:MTEST:TITLe? (see page 600)	<title> ::= a string of up to 128 ASCII characters

**Introduction to :MTEST Commands** Mask testing automatically compares the current displayed waveform with the boundaries of a set of polygons that you define. Any waveform or sample that falls within the boundaries of one or more polygons is recorded as a failure.

### Reporting the Setup

Use :MTEST? to query setup information for the MTEST subsystem.

### Return Format

The following is a sample response from the :MTEST? query. In this case, the query was issued following a \*RST command.

```
:MTES:SOUR CHAN1;ENAB 0;LOCK 1;:MTES:AMAS:SOUR CHAN1;UNIT DIV;XDEL
+2.5000000E-001;YDEL +2.5000000E-001;:MTES:SCAL:X1 +200.000E-06;XDEL
+400.000E-06;Y1 -3.00000E+00;Y2 +3.00000E+00;BIND 0;:MTES:RMOD
FOR;RMOD:TIME +1E+00;WAV 1000;SIGM +6.0E+00;:MTES:RMOD:FACT:STOP
0;PRIN 0;SAVE 0
```

### Example Code

```
' Mask testing commands example.
' -----
Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub Main()
```

```

On Error GoTo VisaComError

' Create the VISA COM I/O resource.
Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488
Set myScope.IO =
    myMgr.Open("USB0::0x0957::0x17A6::US50210029::0::INSTR")
myScope.IO.Clear      ' Clear the interface.

' Make sure oscilloscope is running.
myScope.WriteString ":RUN"

' Set mask test termination conditions.
myScope.WriteString ":MTEST:RMODE SIGMa"
myScope.WriteString ":MTEST:RMODE?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test termination mode: " + strQueryResult

myScope.WriteString ":MTEST:RMODE:SIGMa 4.2"
myScope.WriteString ":MTEST:RMODE:SIGMa?"
varQueryResult = myScope.ReadNumber
Debug.Print "Mask test termination 'test sigma': " + _
    FormatNumber(varQueryResult)

' Use auto-mask to create mask.
myScope.WriteString ":MTEST:AMASK:SOURce CHANnel1"
myScope.WriteString ":MTEST:AMASK:SOURce?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test auto-mask source: " + strQueryResult

myScope.WriteString ":MTEST:AMASK:UNITS DIVisions"
myScope.WriteString ":MTEST:AMASK:UNITS?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test auto-mask units: " + strQueryResult

myScope.WriteString ":MTEST:AMASK:XDELta 0.1"
myScope.WriteString ":MTEST:AMASK:XDELta?"
varQueryResult = myScope.ReadNumber
Debug.Print "Mask test auto-mask X delta: " + _
    FormatNumber(varQueryResult)

myScope.WriteString ":MTEST:AMASK:YDELta 0.1"
myScope.WriteString ":MTEST:AMASK:YDELta?"
varQueryResult = myScope.ReadNumber
Debug.Print "Mask test auto-mask Y delta: " + _
    FormatNumber(varQueryResult)

' Enable "Auto Mask Created" event (bit 10, &H400)
myScope.WriteString "*CLS"
myScope.WriteString ":MTEenable " + CStr(CInt("&H400"))

' Create mask.
myScope.WriteString ":MTEST:AMASK:CREATE"
Debug.Print "Auto-mask created, mask test automatically enabled."

' Set up timeout variables.
Dim lngTimeout As Long      ' Max millisecs to wait.

```

```

Dim lngElapsed As Long
lngTimeout = 60000    ' 60 seconds.

' Wait until mask is created.
lngElapsed = 0
Do While lngElapsed <= lngTimeout
    myScope.WriteString ":OPERRegister:CONDITION?"
    varQueryResult = myScope.ReadNumber
    ' Operation Status Condition Register MTE bit (bit 9, &H200).
    If (varQueryResult And &H200) <> 0 Then
        Exit Do
    Else
        Sleep 100    ' Small wait to prevent excessive queries.
        lngElapsed = lngElapsed + 100
    End If
Loop

' Look for RUN bit = stopped (mask test termination).
lngElapsed = 0
Do While lngElapsed <= lngTimeout
    myScope.WriteString ":OPERRegister:CONDITION?"
    varQueryResult = myScope.ReadNumber
    ' Operation Status Condition Register RUN bit (bit 3, &H8).
    If (varQueryResult And &H8) = 0 Then
        Exit Do
    Else
        Sleep 100    ' Small wait to prevent excessive queries.
        lngElapsed = lngElapsed + 100
    End If
Loop

' Get total waveforms, failed waveforms, and test time.
myScope.WriteString ":MTEST:COUNT:WAVEforms?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test total waveforms: " + strQueryResult

myScope.WriteString ":MTEST:COUNT:FWAVEforms?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test failed waveforms: " + strQueryResult

myScope.WriteString ":MTEST:COUNT:TIME?"
strQueryResult = myScope.ReadString
Debug.Print "Mask test elapsed seconds: " + strQueryResult

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

## :MTEST:ALL

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:ALL <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:ALL command specifies the channel(s) that are included in the mask test:

- ON – All displayed analog channels are included in the mask test.
- OFF – Just the selected source channel is included in the test.

**Query Syntax**    `:MTEST:ENABLE?`

The :MTEST:ENABLE? query returns the current setting.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**    • "Introduction to :MTEST Commands" on page 569

## :MTEST:AMASK:CREate

**N** (see [page 1334](#))

### Command Syntax

`:MTEST :AMASK :CREate`

The :MTEST:AMASK:CREate command automatically constructs a mask around the current selected channel, using the tolerance parameters defined by the :MTEST:AMASK:XDELta, :MTEST:AMASK:YDELta, and :MTEST:AMASK:UNITs commands. The mask only encompasses the portion of the waveform visible on the display, so you must ensure that the waveform is acquired and displayed consistently to obtain repeatable results.

The :MTEST:SOURce command selects the channel and should be set before using this command.

### See Also

- ["Introduction to :MTEST Commands" on page 569](#)
- [":MTEST:AMASK:XDELta" on page 576](#)
- [":MTEST:AMASK:YDELta" on page 577](#)
- [":MTEST:AMASK:UNITs" on page 575](#)
- [":MTEST:AMASK:SOURce" on page 574](#)
- [":MTEST:SOURce" on page 599](#)

### Example Code

- ["Example Code" on page 569](#)

## :MTEST:AMASK:SOURce

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:AMASK:SOURce <source>`

```
<source> ::= CHANnel<n>
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MTEST:AMASK:SOURce command selects the source for the interpretation of the :MTEST:AMASK:XDELta and :MTEST:AMASK:YDELta parameters when :MTEST:AMASK:UNITS is set to CURRent.

When UNITS are CURRent, the XDELta and YDELta parameters are defined in terms of the channel units, as set by the :CHANnel<n>:UNITS command, of the selected source.

Suppose that UNITS are CURRent and that you set SOURce to CHANNEL1, which is using units of volts. Then you can define AMASK:XDELta in terms of volts and AMASK:YDELta in terms of seconds.

This command is the same as the :MTEST:SOURce command.

**Query Syntax**    `:MTEST:AMASK:SOURce?`

The :MTEST:AMASK:SOURce? query returns the currently set source.

**Return Format**

```
<source> ::= CHAN<n>
<n> ::= 1 to (# analog channels) in NR1 format
```

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[:MTEST:AMASK:XDELta](#)" on page 576
- "[:MTEST:AMASK:YDELta](#)" on page 577
- "[:MTEST:AMASK:UNITS](#)" on page 575
- "[:MTEST:SOURce](#)" on page 599

**Example Code**

- "[Example Code](#)" on page 569

## :MTEST:AMASK:UNITS

**N** (see [page 1334](#))

**Command Syntax** :MTEST:AMASK:UNITS <units>

<units> ::= {CURREnt | DIVisions}

The :MTEST:AMASK:UNITS command alters the way the mask test subsystem interprets the tolerance parameters for automasking as defined by :MTEST:AMASK:XDELta and :MTEST:AMASK:YDELta commands.

- CURREnt – the mask test subsystem uses the units as set by the :CHANnel<n>:UNITS command, usually time for  $\Delta X$  and voltage for  $\Delta Y$ .
- DIVisions – the mask test subsystem uses the graticule as the measurement system, so tolerance settings are specified as parts of a screen division. The mask test subsystem maintains separate XDELta and YDELta settings for CURREnt and DIVisions. Thus, XDELta and YDELta are not converted to new values when the UNITS setting is changed.

**Query Syntax** :MTEST:AMASK:UNITS?

The :MTEST:AMASK:UNITS query returns the current measurement units setting for the mask test automask feature.

**Return Format** <units><NL>

<units> ::= {CURR | DIV}

- See Also**
- "[Introduction to :MTEST Commands](#)" on page 569
  - "[:MTEST:AMASK:XDELta](#)" on page 576
  - "[:MTEST:AMASK:YDELta](#)" on page 577
  - "[:CHANnel<n>:UNITS](#)" on page 298
  - "[:MTEST:AMASK:SOURce](#)" on page 574
  - "[:MTEST:SOURce](#)" on page 599

**Example Code**

- "[Example Code](#)" on page 569

## :MTEST:AMASK:XDELta

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:AMASK:XDELta <value>`

`<value> ::= X delta value in NR3 format`

The :MTEST:AMASK:XDELta command sets the tolerance in the X direction around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to horizontal values of the waveform to determine the boundaries of the mask.

The horizontal tolerance value is interpreted based on the setting specified by the :MTEST:AMASK:UNITS command; thus, if you specify 250-E3, the setting for :MTEST:AMASK:UNITS is CURRent, and the current setting specifies time in the horizontal direction, the tolerance will be  $\pm 250$  ms. If the setting for :MTEST:AMASK:UNITS is DIVisions, the same X delta value will set the tolerance to  $\pm 250$  millidivisions, or 1/4 of a division.

**Query Syntax**    `:MTEST:AMASK:XDELta?`

The :MTEST:AMASK:XDELta? query returns the current setting of the  $\Delta X$  tolerance for automasking. If your computer program will interpret this value, it should also request the current measurement system using the :MTEST:AMASK:UNITS query.

**Return Format**    `<value><NL>`

`<value> ::= X delta value in NR3 format`

**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569
- [":MTEST:AMASK:UNITS"](#) on page 575
- [":MTEST:AMASK:YDELta"](#) on page 577
- [":MTEST:AMASK:SOURce"](#) on page 574
- [":MTEST:SOURce"](#) on page 599

**Example Code**

- ["Example Code"](#) on page 569

## :MTEST:AMASK:YDELta

**N** (see [page 1334](#))

**Command Syntax** :MTEST:AMASK:YDELta <value>

<value> ::= Y delta value in NR3 format

The :MTEST:AMASK:YDELta command sets the vertical tolerance around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to vertical values of the waveform to determine the boundaries of the mask.

The vertical tolerance value is interpreted based on the setting specified by the :MTEST:AMASK:UNITS command; thus, if you specify 250-E3, the setting for :MTEST:AMASK:UNITS is CURRent, and the current setting specifies voltage in the vertical direction, the tolerance will be  $\pm 250$  mV. If the setting for :MTEST:AMASK:UNITS is DIVisions, the same Y delta value will set the tolerance to  $\pm 250$  millidivisions, or 1/4 of a division.

**Query Syntax** :MTEST:AMASK:YDELta?

The :MTEST:AMASK:YDELta? query returns the current setting of the  $\Delta Y$  tolerance for automasking. If your computer program will interpret this value, it should also request the current measurement system using the :MTEST:AMASK:UNITS query.

**Return Format** <value><NL>

<value> ::= Y delta value in NR3 format

**See Also** ["Introduction to :MTEST Commands" on page 569](#)

- ["-:MTEST:AMASK:UNITS" on page 575](#)
- ["-:MTEST:AMASK:XDELta" on page 576](#)
- ["-:MTEST:AMASK:SOURce" on page 574](#)
- ["-:MTEST:SOURce" on page 599](#)

**Example Code** ["- Example Code" on page 569](#)

**:MTEST:COUNT:FWAVeforms****N** (see [page 1334](#))**Query Syntax**    `:MTEST:COUNT:FWAVeforms? [CHANnel<n>]``<n> ::= 1 to (# analog channels) in NR1 format`

The :MTEST:COUNT:FWAVeforms? query returns the total number of failed waveforms in the current mask test run. This count is for all regions and all waveforms collected on the channel specified by the optional parameter or collected on the currently specified source channel (:MTEST:SOURce) if there is no parameter.

**Return Format**    `<failed><NL>``<failed> ::= number of failed waveforms in NR1 format.`**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569

- [":MTEST:COUNT:WAVeforms"](#) on page 581
- [":MTEST:COUNT:TIME"](#) on page 580
- [":MTEST:COUNT:RESet"](#) on page 579
- [":MTEST:SOURce"](#) on page 599

**Example Code**

- ["Example Code"](#) on page 569

## :MTEST:COUNt:RESet

**N** (see [page 1334](#))

**Command Syntax** :MTEST:COUNt:RESet

The :MTEST:COUNt:RESet command resets the mask statistics.

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[:MTEST:COUNt:WAVeforms](#)" on page 581
- "[:MTEST:COUNt:FWAVeforms](#)" on page 578
- "[:MTEST:COUNt:TIME](#)" on page 580

**:MTEST:COUNt:TIME**

**N** (see [page 1334](#))

**Query Syntax** `:MTEST:COUNt:TIME?`

The `:MTEST:COUNt:TIME?` query returns the elapsed time in the current mask test run.

**Return Format** `<time><NL>`

`<time>` ::= elapsed seconds in NR3 format.

- See Also**
- "[Introduction to :MTEST Commands](#)" on page 569
  - "[:MTEST:COUNt:WAVEforms](#)" on page 581
  - "[:MTEST:COUNt:FWAveforms](#)" on page 578
  - "[:MTEST:COUNt:RESet](#)" on page 579

**Example Code**

- "[Example Code](#)" on page 569

**:MTEST:COUNt:WAveforms**

**N** (see [page 1334](#))

**Query Syntax** `:MTEST:COUNt:WAveforms?`

The `:MTEST:COUNt:WAveforms?` query returns the total number of waveforms acquired in the current mask test run.

**Return Format** `<count><NL>`

`<count>` ::= number of waveforms in NR1 format.

- See Also**
- "[Introduction to :MTEST Commands](#)" on page 569
  - "[:MTEST:COUNt:FWAVeforms](#)" on page 578
  - "[:MTEST:COUNt:TIME](#)" on page 580
  - "[:MTEST:COUNt:RESet](#)" on page 579

**Example Code**

- "[Example Code](#)" on page 569

**:MTEST:DATA**

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:DATA <mask>`

`<mask> ::= binary block data in IEEE 488.2 # format.`

The :MTEST:DATA command loads a mask from binary block data.

**Query Syntax**    `:MTEST:DATA?`

The :MTEST:DATA? query returns a mask in binary block data format. The format for the data transmission is the # format defined in the IEEE 488.2 specification.

**Return Format**    `<mask><NL>`

`<mask> ::= binary block data in IEEE 488.2 # format`

**See Also**

- [":SAVE:MASK\[:START\]" on page 705](#)
- [":RECall:MASK\[:START\]" on page 689](#)

## :MTEST:DELetE

**N** (see [page 1334](#))

**Command Syntax** :MTEST:DELetE

The :MTEST:DELetE command clears the currently loaded mask.

**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569
- [":MTEST:AMASK:CREate"](#) on page 573

## :MTEST:ENABLE

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:ENABLE <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:ENABLE command enables or disables the mask test features.

- ON – Enables the mask test features.
- OFF – Disables the mask test features.

**Query Syntax**    `:MTEST:ENABLE?`

The :MTEST:ENABLE? query returns the current state of mask test features.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**    · "Introduction to :MTEST Commands" on page 569

## :MTEST:LOCK

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:LOCK <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:LOCK command enables or disables the mask lock feature:

- ON – Locks a mask to the SOURce. As the vertical or horizontal scaling or position of the SOURce changes, the mask is redrawn accordingly.
- OFF – The mask is static and does not move.

**Query Syntax**    `:MTEST:LOCK?`

The :MTEST:LOCK? query returns the current mask lock setting.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569

- "[":MTEST:SOURce](#)" on page 599

## :MTESt:RMODE

**N** (see [page 1334](#))

**Command Syntax**    `:MTESt:RMODE <rmode>`

`<rmode> ::= {FORever | SIGMa | TIME | WAVEforms}`

The :MTESt:RMODE command specifies the termination conditions for the mask test:

- FORever – the mask test runs until it is turned off.
- SIGMa – the mask test runs until the Sigma level is reached. This level is set by the "[:MTESt:RMODE:SIGMa](#)" on page 591 command.
- TIME – the mask test runs for a fixed amount of time. The amount of time is set by the "[:MTESt:RMODE:TIME](#)" on page 592 command.
- WAVEforms – the mask test runs until a fixed number of waveforms are acquired. The number of waveforms is set by the "[:MTESt:RMODE:WAVEforms](#)" on page 593 command.

**Query Syntax**    `:MTESt:RMODE?`

The :MTESt:RMODE? query returns the currently set termination condition.

**Return Format**    `<rmode><NL>`

`<rmode> ::= {FOR | SIGM | TIME | WAV}`

**See Also**

- "[Introduction to :MTESt Commands](#)" on page 569
- "[:MTESt:RMODE:SIGMa](#)" on page 591
- "[:MTESt:RMODE:TIME](#)" on page 592
- "[:MTESt:RMODE:WAVEforms](#)" on page 593

**Example Code**

- "[Example Code](#)" on page 569

## :MTEST:RMODE:FACTion:MEASure

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:RMODE:FACTion:MEASure <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:RMODE:FACTion:MEASure command sets measuring only mask failures on or off.

When ON, measurements and measurement statistics run only on waveforms that contain a mask violation; passing waveforms do not affect measurements and measurement statistics.

This mode is not available when the acquisition mode is set to Averaging.

**Query Syntax**    `:MTEST:RMODE:FACTion:MEASure?`

The :MTEST:RMODE:FACTion:MEASure? query returns the current mask failure measure setting.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:RMODE:FACTion:PRINT](#)" on page 588
- "[":MTEST:RMODE:FACTion:SAVE](#)" on page 589
- "[":MTEST:RMODE:FACTion:STOP](#)" on page 590

**:MTESt:RMODE:FACTion:PRINt****N** (see [page 1334](#))

**Command Syntax**    `:MTESt:RMODE:FACTion:PRINt <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTESt:RMODE:FACTion:PRINt command sets printing on mask failures on or off.

**NOTE**

Setting :MTESt:RMODE:FACTion:PRINt ON automatically sets :MTESt:RMODE:FACTion:SAVE OFF.

---

See [Chapter 19](#), “:HARDcopy Commands,” starting on page 417 for more information on setting the hardcopy device and formatting options.

**Query Syntax**    `:MTESt:RMODE:FACTion:PRINt?`

The :MTESt:RMODE:FACTion:PRINt? query returns the current mask failure print setting.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**

- ["Introduction to :MTESt Commands"](#) on page 569
- [":MTESt:RMODE:FACTion:MEASure"](#) on page 587
- [":MTESt:RMODE:FACTion:SAVE"](#) on page 589
- [":MTESt:RMODE:FACTion:STOP"](#) on page 590

## :MTEST:RMODE:FACTion:SAVE

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:RMODE:FACTion:SAVE <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:RMODE:FACTion:SAVE command sets saving on mask failures on or off.

### NOTE

Setting :MTEST:RMODE:FACTion:SAVE ON automatically sets :MTEST:RMODE:FACTion:PRINT OFF.

See [Chapter 28](#), “:SAVE Commands,” starting on page 693 for more information on save options.

**Query Syntax**    `:MTEST:RMODE:FACTion:SAVE?`

The :MTEST:RMODE:FACTion:SAVE? query returns the current mask failure save setting.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569
- [":MTEST:RMODE:FACTion:MEASure"](#) on page 587
- [":MTEST:RMODE:FACTion:PRINT"](#) on page 588
- [":MTEST:RMODE:FACTion:STOP"](#) on page 590

**:MTEST:RMODE:FACTion:STOP****N** (see [page 1334](#))

**Command Syntax**    `:MTEST:RMODE:FACTion:STOP <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:RMODE:FACTion:STOP command sets stopping on a mask failure on or off. When this setting is ON and a mask violation is detected, the mask test is stopped and the acquisition system is stopped.

**Query Syntax**    `:MTEST:RMODE:FACTion:STOP?`

The :MTEST:RMODE:FACTion:STOP? query returns the current mask failure stop setting.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:RMODE:FACTion:MEASure](#)" on page 587
- "[":MTEST:RMODE:FACTion:PRINT](#)" on page 588
- "[":MTEST:RMODE:FACTion:SAVE](#)" on page 589

**:MTEST:RMODE:SIGMA**

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:RMODE:SIGMA <level>`

`<level> ::= from 0.1 to 9.3 in NR3 format`

When the :MTEST:RMODE command is set to SIGMa, the :MTEST:RMODE:SIGMA command sets the *test sigma* level to which a mask test runs. *Test sigma* is the best achievable process sigma, assuming no failures. (*Process sigma* is calculated using the number of failures per test.) The *test sigma* level indirectly specifies the number of waveforms that must be tested (in order to reach the sigma level).

**Query Syntax**    `:MTEST:RMODE:SIGMA?`

The :MTEST:RMODE:SIGMA? query returns the current Sigma level setting.

**Return Format**    `<level><NL>`

`<level> ::= from 0.1 to 9.3 in NR3 format`

**See Also**

- "Introduction to :MTEST Commands" on page 569
- ":MTEST:RMODE" on page 586

**Example Code**

- "[Example Code](#)" on page 569

**:MTESt:RMODE:TIME**

**N** (see [page 1334](#))

**Command Syntax**    `:MTESt:RMODE:TIME <seconds>`

`<seconds> ::= from 1 to 86400 in NR3 format`

When the :MTESt:RMODE command is set to TIME, the :MTESt:RMODE:TIME command sets the number of seconds for a mask test to run.

**Query Syntax**    `:MTESt:RMODE:TIME?`

The :MTESt:RMODE:TIME? query returns the number of seconds currently set.

**Return Format**    `<seconds><NL>`

`<seconds> ::= from 1 to 86400 in NR3 format`

**See Also**

- "Introduction to :MTESt Commands" on page 569
- "[:MTESt:RMODE](#)" on page 586

## :MTEST:RMODE:WAVEforms

**N** (see [page 1334](#))

**Command Syntax** :MTEST:RMODE:WAVEforms <count>

<count> ::= number of waveforms in NR1 format  
from 1 to 2,000,000,000

When the :MTEST:RMODE command is set to WAVEforms, the :MTEST:RMODE:WAVEforms command sets the number of waveform acquisitions that are mask tested.

**Query Syntax** :MTEST:RMODE:WAVEforms?

The :MTEST:RMODE:WAVEforms? query returns the number of waveforms currently set.

**Return Format** <count><NL>

<count> ::= number of waveforms in NR1 format  
from 1 to 2,000,000,000

**See Also** • "[Introduction to :MTEST Commands](#)" on page 569  
• "[":MTEST:RMODE"](#) on page 586

## :MTESt:SCALe:BIND

**N** (see [page 1334](#))

**Command Syntax**    `:MTESt:SCALe:BIND <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTESt:SCALe:BIND command enables or disables Bind 1 & 0 Levels (Bind -1 & 0 Levels for inverted masks) control:

- ON –

If the Bind 1 & 0 Levels control is enabled, the 1 Level and the 0 Level controls track each other. Adjusting either the 1 Level or the 0 Level control shifts the position of the mask up or down without changing its size.

If the Bind -1 & 0 Levels control is enabled, the -1 Level and the 0 Level controls track each other. Adjusting either the -1 Level or the 0 Level control shifts the position of the mask up or down without changing its size.

- OFF –

If the Bind 1 & 0 Levels control is disabled, adjusting either the 1 Level or the 0 Level control changes the vertical height of the mask.

If the Bind -1 & 0 Levels control is disabled, adjusting either the -1 Level or the 0 Level control changes the vertical height of the mask.

**Query Syntax**    `:MTESt:SCALe:BIND?`

The :MTESt:SCALe:BIND? query returns the value of the Bind 1&0 control (Bind -1&0 for inverted masks).

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- "[Introduction to :MTESt Commands](#)" on page 569
- "[":MTESt:SCALe:X1](#)" on page 595
- "[":MTESt:SCALe:XDELta](#)" on page 596
- "[":MTESt:SCALe:Y1](#)" on page 597
- "[":MTESt:SCALe:Y2](#)" on page 598

## :MTEST:SCALe:X1

**N** (see [page 1334](#))

**Command Syntax** :MTEST:SCALe:X1 <x1\_value>

<x1\_value> ::= X1 value in NR3 format

The :MTEST:SCALe:X1 command defines where X=0 in the base coordinate system used for mask testing. The other X-coordinate is defined by the :MTEST:SCALe:XDELta command. Once the X1 and XDELta coordinates are set, all X values of vertices in the mask regions are defined with respect to this value, according to the equation:

$$X = (X * \Delta X) + X1$$

Thus, if you set X1 to 100 ms, and XDELta to 100 ms, an X value of 0.100 is a vertex at 110 ms.

The oscilloscope uses this equation to normalize vertices. This simplifies reprogramming to handle different data rates. For example, if you halve the period of the waveform of interest, you need only to adjust the XDELta value to set up the mask for the new waveform.

The X1 value is a time value specifying the location of the X1 coordinate, which will then be treated as X=0 for mask regions coordinates.

**Query Syntax** :MTEST:SCALe:X1?

The :MTEST:SCALe:X1? query returns the current X1 coordinate setting.

**Return Format** <x1\_value><NL>

<x1\_value> ::= X1 value in NR3 format

- See Also**
- ["Introduction to :MTEST Commands" on page 569](#)
  - [":MTEST:SCALe:BIND" on page 594](#)
  - [":MTEST:SCALe:XDELta" on page 596](#)
  - [":MTEST:SCALe:Y1" on page 597](#)
  - [":MTEST:SCALe:Y2" on page 598](#)

## :MTEST:SCALe:XDELta

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:SCALe:XDELta <xdelta_value>`

`<xdelta_value> ::= X delta value in NR3 format`

The :MTEST:SCALe:XDELta command defines the position of the X2 marker with respect to the X1 marker. In the mask test coordinate system, the X1 marker defines where X=0; thus, the X2 marker defines where X=1.

Because all X vertices of the regions defined for mask testing are normalized with respect to X1 and  $\Delta X$ , redefining  $\Delta X$  also moves those vertices to stay in the same locations with respect to X1 and  $\Delta X$ . Thus, in many applications, it is best if you define XDELta as a pulse width or bit period. Then, a change in data rate without corresponding changes in the waveform can easily be handled by changing  $\Delta X$ .

The X-coordinate of polygon vertices is normalized using this equation:

$$X = (X * \Delta X) + X1$$

The X delta value is a time value specifying the distance of the X2 marker with respect to the X1 marker.

For example, if the period of the waveform you wish to test is 1 ms, setting  $\Delta X$  to 1 ms ensures that the waveform's period is between the X1 and X2 markers.

**Query Syntax**    `:MTEST:SCALe:XDELta?`

The :MTEST:SCALe:XDELta? query returns the current value of  $\Delta X$ .

**Return Format**    `<xdelta_value><NL>`

`<xdelta_value> ::= X delta value in NR3 format`

**See Also**

- ["Introduction to :MTEST Commands" on page 569](#)
- [":MTEST:SCALe:BIND" on page 594](#)
- [":MTEST:SCALe:X1" on page 595](#)
- [":MTEST:SCALe:Y1" on page 597](#)
- [":MTEST:SCALe:Y2" on page 598](#)

## :MTEST:SCALe:Y1

**N** (see [page 1334](#))

**Command Syntax** :MTEST:SCALe:Y1 <y1\_value>

<y1\_value> ::= Y1 value in NR3 format

The :MTEST:SCALe:Y1 command defines where Y=0 in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries set by SCALe:Y1 and SCALe:Y2 according to the equation:

$$Y = (Y * (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1 V, a Y value of 0.100 in a vertex is at 190 mV.

The Y1 value is a voltage value specifying the point at which Y=0.

**Query Syntax** :MTEST:SCALe:Y1?

The :MTEST:SCALe:Y1? query returns the current setting of the Y1 marker.

**Return Format** <y1\_value><NL>

<y1\_value> ::= Y1 value in NR3 format

- See Also**
- "[Introduction to :MTEST Commands](#)" on page 569
  - "[":MTEST:SCALe:BIND](#)" on page 594
  - "[":MTEST:SCALe:X1](#)" on page 595
  - "[":MTEST:SCALe:XDELta](#)" on page 596
  - "[":MTEST:SCALe:Y2](#)" on page 598

## :MTEST:SCALe:Y2

**N** (see [page 1334](#))

**Command Syntax**    `:MTEST:SCALe:Y2 <y2_value>`

`<y2_value> ::= Y2 value in NR3 format`

The :MTEST:SCALe:Y2 command defines the Y2 marker in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries defined by SCALe:Y1 and SCALe:Y2 according to the following equation:

$$Y = (Y * (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1 V, a Y value of 0.100 in a vertex is at 190 mV.

The Y2 value is a voltage value specifying the location of the Y2 marker.

**Query Syntax**    `:MTEST:SCALe:Y2?`

The :MTEST:SCALe:Y2? query returns the current setting of the Y2 marker.

**Return Format**    `<y2_value><NL>`

`<y2_value> ::= Y2 value in NR3 format`

**See Also**

- ["Introduction to :MTEST Commands" on page 569](#)
- [":MTEST:SCALe:BIND" on page 594](#)
- [":MTEST:SCALe:X1" on page 595](#)
- [":MTEST:SCALe:XDELta" on page 596](#)
- [":MTEST:SCALe:Y1" on page 597](#)

## :MTEST:SOURce

**N** (see [page 1334](#))

**Command Syntax** :MTEST:SOURce <source>

```
<source> ::= CHANnel<n>
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MTEST:SOURce command selects the channel which is configured by the commands contained in a mask file when it is loaded.

**Query Syntax** :MTEST:SOURce?

The :MTEST:SOURce? query returns the channel which is configured by the commands contained in the current mask file.

**Return Format** <source><NL>

```
<source> ::= {CHAN<n> | NONE}
<n> ::= 1 to (# analog channels) in NR1 format
```

**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569
- [":MTEST:AMASK:SOURce"](#) on page 574

## :MTEST:TITLE

**N** (see [page 1334](#))

**Query Syntax**    `:MTEST:TITLE?`

The `:MTEST:TITLE?` query returns the mask title which is a string of up to 128 characters. The title is displayed in the mask test dialog box and mask test tab when a mask file is loaded.

**Return Format**    `<title><NL>`

`<title>` ::= a string of up to 128 ASCII characters.

**See Also**    · "Introduction to [:MTEST Commands](#)" on page 569

## 25 :POD Commands

Control all oscilloscope functions associated with groups of digital channels. See "[Introduction to :POD<n> Commands](#)" on page 601.

**Table 106:**:POD<n> Commands Summary

Command	Query	Options and Query Returns
:POD<n>:DISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 603</a> )	:POD<n>:DISPlay? (see <a href="#">page 603</a> )	{0   1} <n> ::= 1-2 in NR1 format
:POD<n>:SIZE <value> (see <a href="#">page 604</a> )	:POD<n>:SIZE? (see <a href="#">page 604</a> )	<value> ::= {SMALL   MEDIUM   LARGe}
:POD<n>:THreshold <type>[suffix] (see <a href="#">page 605</a> )	:POD<n>:THreshold? (see <a href="#">page 605</a> )	<n> ::= 1-2 in NR1 format <type> ::= {CMOS   ECL   TTL   <user defined value>} <user defined value> ::= value in NR3 format [suffix] ::= {V   mV   uV }

**Introduction to :POD<n> Commands**    <n> ::= {1 | 2}

The POD subsystem commands control the viewing and threshold of groups of digital channels.

POD1 ::= D0-D7

POD2 ::= D8-D15

**NOTE**

These commands are only valid for the MSO models.

---

### Reporting the Setup

Use :POD1? or :POD2? to query setup information for the POD subsystem.

### Return Format

The following is a sample response from the :POD1? query. In this case, the query was issued following a \*RST command.

```
:POD1:DISP 0;THR +1.40E+00
```

## :POD<n>:DISPlay

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:POD&lt;n&gt;:DISPlay &lt;display&gt;</code>
	<code>&lt;display&gt; ::= {{1   ON}   {0   OFF}}</code>
	<code>&lt;n&gt; ::= An integer, 1 or 2, is attached as a suffix to the command and defines the group of channels that are affected by the command.</code>
	<code>POD1 ::= D0-D7</code>
	<code>POD2 ::= D8-D15</code>
	The :POD<n>:DISPlay command turns displaying of the specified group of channels on or off.

### NOTE

This command is only valid for the MSO models.

### Query Syntax

`:POD<n>:DISPlay?`

The :POD<n>:DISPlay? query returns the current display setting of the specified group of channels.

### Return Format

`<display><NL>`

`<display> ::= {0 | 1}`

### See Also

- "[Introduction to :POD<n> Commands](#)" on page 601
- "[:DIGItal<d>:DISPlay](#)" on page 323
- "[:CHANnel<n>:DISPlay](#)" on page 284
- "[:VIEW](#)" on page 242
- "[:BLANK](#)" on page 214
- "[:STATus](#)" on page 239

## :POD<n>:SIZE

**N** (see [page 1334](#))

**Command Syntax**    `:POD<n>:SIZE <value>`

`<n>` ::= An integer, 1 or 2, is attached as a suffix to the command and defines the group of channels that are affected by the command.

`POD1` ::= D0-D7

`POD2` ::= D8-D15

`<value>` ::= {SMALL | MEDium | LARGe}

The :POD<n>:SIZE command specifies the size of digital channels on the display. Sizes are set for all pods. Therefore, if you set the size on pod 1 (for example), the same size is set on pod 2 as well.

### NOTE

This command is only valid for the MSO models.

**Query Syntax**    `:POD<n>:SIZE?`

The :POD<n>:SIZE? query returns the digital channels size setting.

**Return Format**    `<size_value><NL>`

`<size_value>` ::= {SMAL | MED | LARG}

**See Also**

- "Introduction to :POD<n> Commands" on page 601
- "[:DIGital<d>:SIZE](#)" on page 326
- "[:DIGital<d>:POSITION](#)" on page 325

## :POD<n>:THreshold

**N** (see [page 1334](#))

### Command Syntax :POD<n>:THreshold <type>[<suffix>]

<n> ::= An integer, 1 or 2, is attached as a suffix to the command and defines the group of channels that are affected by the command.

<type> ::= {CMOS | ECL | TTL | <user defined value>}

<user defined value> ::= -8.00 to +8.00 in NR3 format

<suffix> ::= {V | mV | uV}

POD1 ::= D0-D7

POD2 ::= D8-D15

TTL ::= 1.4V

CMOS ::= 2.5V

ECL ::= -1.3V

The :POD<n>:THreshold command sets the threshold for the specified group of channels. The threshold is used for triggering purposes and for displaying the digital data as high (above the threshold) or low (below the threshold).

### NOTE

This command is only valid for the MSO models.

### Query Syntax :POD<n>:THreshold?

The :POD<n>:THreshold? query returns the threshold value for the specified group of channels.

### Return Format <threshold><NL>

<threshold> ::= Floating point number in NR3 format

### See Also

- "[Introduction to :POD<n> Commands](#)" on page 601
- "[:DIGItal<d>:THreshold](#)" on page 327
- "[:TRIGger\[:EDGE\]:LEVel](#)" on page 1073

### Example Code

```
' THRESHOLD - This command is used to set the voltage threshold for
' the waveforms. There are three preset values (TTL, CMOS, and ECL)
' and you can also set a user-defined threshold value between
' -8.0 volts and +8.0 volts.
'
' In this example, we set channels 0-7 to CMOS, then set channels
' 8-15 to a user-defined 2.0 volts, and then set the external trigger
' to TTL. Of course, you only need to set the thresholds for the
' channels you will be using in your program.
```

```
' Set channels 0-7 to CMOS threshold.  
myScope.WriteString ":POD1:THRESHOLD CMOS"  
  
' Set channels 8-15 to 2.0 volts.  
myScope.WriteString ":POD2:THRESHOLD 2.0"  
  
' Set external channel to TTL threshold (short form).  
myScope.WriteString ":TRIG:LEV TTL,EXT"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## 26 :POWer Commands

These :POWer commands are available when the DSOX3PWR power measurements and analysis application is licensed and enabled.

**Table 107:**POWer Commands Summary

Command	Query	Options and Query Returns
:POWer:CLResponse:APP Ly (see <a href="#">page 613</a> )	n/a	n/a
:POWer:CLResponse:FRE Quency:START <value>[suffix] (see <a href="#">page 614</a> )	:POWer:CLResponse:FRE Quency:START? (see <a href="#">page 614</a> )	<value> ::= {20   100   1000   10000   100000   1000000   10000000} [suffix] ::= {Hz   kHz   MHz}
:POWer:CLResponse:FRE Quency:STOP <value>[suffix] (see <a href="#">page 615</a> )	:POWer:CLResponse:FRE Quency:STOP? (see <a href="#">page 615</a> )	<value> ::= {100   1000   10000   100000   1000000   10000000   20000000} [suffix] ::= {Hz   kHz   MHz}
:POWer:CLResponse:VIE W <view_type> (see <a href="#">page 616</a> )	:POWer:CLResponse:VIE W? (see <a href="#">page 616</a> )	<view_type> ::= {GAIN   PHASE}
:POWer:CLResponse:YMA Ximum <value> (see <a href="#">page 617</a> )	:POWer:CLResponse:YMA Ximum? (see <a href="#">page 617</a> )	<value> ::= dB value in multiples of 10 from -110 to 120
:POWer:CLResponse:YMI Nimum <value> (see <a href="#">page 618</a> )	:POWer:CLResponse:YMM INnum? (see <a href="#">page 618</a> )	<value> ::= dB value in multiples of 10 from -120 to 110
:POWer:DESkew (see <a href="#">page 619</a> )	n/a	n/a
:POWer:EFFiciency:APP Ly (see <a href="#">page 620</a> )	n/a	n/a
:POWer:EFFiciency:TYP E <type> (see <a href="#">page 621</a> )	:POWer:EFFiciency:TYP E? (see <a href="#">page 621</a> )	<type> ::= {DCDC   DCAC   ACDC   ACAC}

**Table 107:**POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:ENABLE {{0   OFF}   {1   ON}} (see page 622)	:POWer:ENABLE? (see page 622)	{0   1}
:POWer:HARMonics:APPLy (see page 623)	n/a	n/a
n/a	:POWer:HARMonics:DATA? (see page 624)	<binary_block> ::= comma-separated data with newlines at the end of each row
:POWer:HARMonics:DISPLAY <display> (see page 625)	:POWer:HARMonics:DISPLAY? (see page 625)	<display> ::= {TABLE   BAR   OFF}
n/a	:POWer:HARMonics:FAILCOUNT? (see page 626)	<count> ::= integer in NR1 format
:POWer:HARMonics:LINE <frequency> (see page 627)	:POWer:HARMonics:LINE? (see page 627)	<frequency> ::= {F50   F60   F400}
n/a	:POWer:HARMonics:POWERFACtor? (see page 628)	<value> ::= Class C power factor in NR3 format
:POWer:HARMonics:RPOWER <source> (see page 629)	:POWer:HARMonics:RPOWER? (see page 629)	<source> ::= {MEASured   USER}
:POWer:HARMonics:RPOWER:USER <value> (see page 630)	:POWer:HARMonics:RPOWER:USER? (see page 630)	<value> ::= Watts from 1.0 to 600.0 in NR3 format
n/a	:POWer:HARMonics:RUNCOUNT? (see page 631)	<count> ::= integer in NR1 format
:POWer:HARMonics:STANDARD <class> (see page 632)	:POWer:HARMonics:STANDARD? (see page 632)	<class> ::= {A   B   C   D}
n/a	:POWer:HARMonics:STATUS? (see page 633)	<status> ::= {PASS   FAIL   UNTESTED}
n/a	:POWer:HARMonics:THD? (see page 634)	<value> ::= Total Harmonics Distortion in NR3 format
:POWer:INRUSH:APPLY (see page 635)	n/a	n/a
:POWer:INRUSH:EXIT (see page 636)	n/a	n/a

**Table 107:** POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:INRush:NEXT (see <a href="#">page 637</a> )	n/a	n/a
:POWER:MODulation:APP Ly (see <a href="#">page 638</a> )	n/a	n/a
:POWER:MODulation:SOU Rce <source> (see <a href="#">page 639</a> )	:POWER:MODulation:SOU Rce? (see <a href="#">page 639</a> )	<source> ::= {V   I}
:POWER:MODulation:TYP E <modulation> (see <a href="#">page 640</a> )	:POWER:MODulation:TYP E? (see <a href="#">page 640</a> )	<modulation> ::= {VAverage   ACRMs   VRATio   PERiod   FREQuency   PWIDith   NWIDth   DUTYcycle   RISetime   FALLtime}
:POWER:ONOFF:APPLY (see <a href="#">page 641</a> )	n/a	n/a
:POWER:ONOFF:EXIT (see <a href="#">page 642</a> )	n/a	n/a
:POWER:ONOFF:NEXT (see <a href="#">page 643</a> )	n/a	n/a
:POWER:ONOFF:TEST {{0   OFF}   {1   ON}} (see <a href="#">page 644</a> )	:POWER:ONOFF:TEST? (see <a href="#">page 644</a> )	{0   1}
:POWER:PSRR:APPLY (see <a href="#">page 645</a> )	n/a	n/a
:POWER:PSRR:FREQuency :MAXimum <value>[suffix] (see <a href="#">page 646</a> )	:POWER:PSRR:FREQuency :MAXimum? (see <a href="#">page 646</a> )	<value> ::= {10   100   1000   10000   100000   1000000   10000000   20000000} [suffix] ::= {Hz   kHz   MHz}
:POWER:PSRR:FREQuency :MINimum <value>[suffix] (see <a href="#">page 647</a> )	:POWER:PSRR:FREQuency :MINimum? (see <a href="#">page 647</a> )	<value> ::= {1   10   100   1000   10000   100000   1000000   10000000} [suffix] ::= {Hz   kHz   MHz}
:POWER:PSRR:RMAXimum <value> (see <a href="#">page 648</a> )	:POWER:PSRR:RMAXimum? (see <a href="#">page 648</a> )	<value> ::= Maximum ratio value in NR1 format
:POWER:QUALity:APPLY (see <a href="#">page 649</a> )	n/a	n/a
:POWER:RIPPLE:APPLY (see <a href="#">page 650</a> )	n/a	n/a
:POWER:SIGNals:AUTOse tup <analysis> (see <a href="#">page 651</a> )	n/a	<analysis> ::= {HARMonics   EFFiciency   RIPPLE   MODulation   QUALity   SLEW   SWITch   RDSVce}

**Table 107:** POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:SIGNals:CYCLes :HARMonics <count> (see page 652)	:POWER:SIGNals:CYCLes :HARMonics? (see page 652)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWER:SIGNals:CYCLes :QUALity <count> (see page 653)	:POWER:SIGNals:CYCLes :QUALity? (see page 653)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWER:SIGNals:DURati on:EFFiciency <value>[suffix] (see page 654)	:POWER:SIGNals:DURati on:EFFiciency? (see page 654)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:MODulation <value>[suffix] (see page 655)	:POWER:SIGNals:DURati on:MODulation? (see page 655)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:ONOFF:OFF <value>[suffix] (see page 656)	:POWER:SIGNals:DURati on:ONOFF:OFF? (see page 656)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:ONOFF:ON <value>[suffix] (see page 657)	:POWER:SIGNals:DURati on:ONOFF:ON? (see page 657)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:RIPPle <value>[suffix] (see page 658)	:POWER:SIGNals:DURati on:RIPPle? (see page 658)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:DURati on:TRANSient <value>[suffix] (see page 659)	:POWER:SIGNals:DURati on:TRANSient? (see page 659)	<value> ::= value in NR3 format [suffix] ::= {s   ms   us   ns}
:POWER:SIGNals:IEXPec ted <value>[suffix] (see page 660)	:POWER:SIGNals:IEXPec ted? (see page 660)	<value> ::= Expected current value in NR3 format [suffix] ::= {A   mA}
:POWER:SIGNals:OVERsh oot <percent> (see page 661)	:POWER:SIGNals:OVERsh oot? (see page 661)	<percent> ::= percent of overshoot value in NR1 format [suffix] ::= {V   mV}}
:POWER:SIGNals:VMAXim um:INRush <value>[suffix] (see page 662)	:POWER:SIGNals:VMAXim um:INRush? (see page 662)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}

**Table 107:** POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:SIGNals:VMAXimum:ONOFF:OFF <value>[suffix] (see page 663)	:POWER:SIGNals:VMAXimum:ONOFF:OFF? (see page 663)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VMAXimum:ONOFF:ON <value>[suffix] (see page 664)	:POWER:SIGNals:VMAXimum:ONOFF:ON? (see page 664)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:ONOFF:OFF <value>[suffix] (see page 665)	:POWER:SIGNals:VSTeady:ONOFF:OFF? (see page 665)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:ONOFF:ON <value>[suffix] (see page 666)	:POWER:SIGNals:VSTeady:ONOFF:ON? (see page 666)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:VSTeady:TRANSient <value>[suffix] (see page 667)	:POWER:SIGNals:VSTeady:TRANSient? (see page 667)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V   mV}
:POWER:SIGNals:SOURce:CURRent<i> <source> (see page 668)	:POWER:SIGNals:SOURce:CURRent<i>? (see page 668)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:SIGNals:SOURce:VOLTage<i> <source> (see page 669)	:POWER:SIGNals:SOURce:VOLTage<i>? (see page 669)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:SLEW:APPLY (see page 670)	n/a	n/a
:POWER:SLEW:SOURce <source> (see page 671)	:POWER:SLEW:SOURce? (see page 671)	<source> ::= {V   I}
:POWER:SWITch:APPLY (see page 672)	n/a	n/a
:POWER:SWITch:CONDUCTion <conduction> (see page 673)	:POWER:SWITch:CONDUCTion? (see page 673)	<conduction> ::= {WAVEform   RDS   VCE}
:POWER:SWITch:IREFerence <percent> (see page 674)	:POWER:SWITch:IREFerence? (see page 674)	<percent> ::= percent in NR1 format

**Table 107:**POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:SWITch:RDS <value>[suffix] (see page 675)	:POWer:SWITch:RDS? (see page 675)	<value> ::= Rds(on) value in NR3 format [suffix] ::= {OHM   mOHM}
:POWer:SWITch:VCE <value>[suffix] (see page 676)	:POWer:SWITch:VCE? (see page 676)	<value> ::= Vce(sat) value in NR3 format [suffix] ::= {V   mV}
:POWer:SWITch:VREFerence <percent> (see page 677)	:POWer:SWITch:VREFerence? (see page 677)	<percent> ::= percent in NR1 format
:POWer:TRANSient:APPLy (see page 678)	n/a	n/a
:POWer:TRANSient:EXIT (see page 679)	n/a	n/a
:POWer:TRANSient:IINITial <value>[suffix] (see page 680)	:POWer:TRANSient:IINITial? (see page 680)	<value> ::= Initial current value in NR3 format [suffix] ::= {A   mA}
:POWer:TRANSient:INEW <value>[suffix] (see page 681)	:POWer:TRANSient:INEW? (see page 681)	<value> ::= New current value in NR3 format [suffix] ::= {A   mA}
:POWer:TRANSient:NEXT (see page 682)	n/a	n/a

## :POWer:CLResponse:APPLy



(see [page 1334](#))

### Command Syntax

`:POWer:CLResponse:APPLy`

The :POWer:CLResponse:APPLy command performs the control loop response (Bode) analysis to help you determine the margin of a control loop.

A Bode plot measurement plots gain or phase (selected by the :POWer:CLResponse:VIEW command) as a function of frequency.

This control loop response analysis requires an input sine wave (from the oscilloscope's waveform generator, Vi) be swept from a low to a high frequency while measuring Vi and Vo RMS voltages at each step frequency, using two channels of the oscilloscope.

For a gain plot, gain (A, in dB units) at each step frequency is computed as  $20\log(V_o/V_i)$  and plotted using a math function waveform.

For a phase plot, the phase difference between the channels is measured at each step frequency. Phase measurements and plots are only possible if the input and output waveforms exceed 1 division peak-to-peak (>1 mVpp).

It takes some time for the frequency sweep analysis to complete. You can query bit 0 of the Standard Event Status Register (\*ESR?) to find out when the analysis is complete.

### See Also

- "[":POWer:CLResponse:FREQuency:STARt](#)" on page 614
- "[":POWer:CLResponse:FREQuency:STOP](#)" on page 615
- "[":POWer:CLResponse:VIEW](#)" on page 616
- "[":POWer:CLResponse:YMAXimum](#)" on page 617
- "[":POWer:CLResponse:YMINimum](#)" on page 618
- "[":\\*ESR \(Standard Event Status Register\)](#)" on page 184

## :POWer:CLResponse:FREQuency:STARt

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:CLResponse:FREQuency:STARt <value>[suffix]`

`<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}`

`[suffix] ::= {Hz | kHz | MHz}`

The :POWer:CLResponse:FREQuency:STARt command sets the frequency sweep start value. The control loop response analysis is displayed on a log scale Bode plot, so you can select from decade values in addition to the minimum frequency of 20 Hz.

**Query Syntax**    `:POWer:CLResponse:FREQuency:STARt?`

The :POWer:CLResponse:FREQuency:STARt? query returns the frequency sweep start setting.

**Return Format**    `<value><NL>`

`<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}`

**See Also**

- "[":POWer:CLResponse:APPLy](#)" on page 613
- "[":POWer:CLResponse:FREQuency:STOP](#)" on page 615
- "[":POWer:CLResponse:VIEW](#)" on page 616
- "[":POWer:CLResponse:YMAXimum](#)" on page 617
- "[":POWer:CLResponse:YMINimum](#)" on page 618

## :POWer:CLResponse:FREQuency:STOP

**N** (see [page 1334](#))

**Command Syntax**

```
:POWer:CLResponse:FREQuency:STOP <value>[suffix]
<value> ::= {100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 20000000}
[suffix] ::= {Hz | kHz | MHz}
```

The :POWer:CLResponse:FREQuency:STOP command sets the frequency sweep stop value. The control loop response analysis is displayed on a log scale Bode plot, so you can select from decade values in addition to the maximum frequency of 20 MHz.

**Query Syntax**

```
:POWer:CLResponse:FREQuency:STOP?
```

The :POWer:CLResponse:FREQuency:STOP? query returns the frequency sweep stop setting.

**Return Format**

```
<value><NL>
<value> ::= {100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 20000000}
```

**See Also**

- "[:POWer:CLResponse:APPLy](#)" on page 613
- "[:POWer:CLResponse:FREQuency:START](#)" on page 614
- "[:POWer:CLResponse:VIEW](#)" on page 616
- "[:POWer:CLResponse:YMAXimum](#)" on page 617
- "[:POWer:CLResponse:YMINimum](#)" on page 618

## :POWer:CLResponse:VIEW

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:CLResponse:VIEW <view_type>`  
`<value> ::= {GAIN | PHASE}`

The :POWer:CLResponse:VIEW command selects which plot to display. The value can only be adjusted when the selected Analysis is "Control Loop Response (Bode)".

Phase measurements and plots are only possible if the input and output waveforms exceed 1 division peak-to-peak (>1 mVpp).

**Query Syntax**    `:POWer:CLResponse:VIEW?`

The :POWer:CLResponse:VIEW? query returns the view type setting.

**Return Format**    `<view_type><NL>`  
`<view_type> ::= {GAIN | PHASE | NONE}`

NONE is not selectable and is returned when there is no prior analysis to plot.

**See Also**

- "[:POWer:CLResponse:APPLy](#)" on page 613
- "[:POWer:CLResponse:FREQuency:STARt](#)" on page 614
- "[:POWer:CLResponse:FREQuency:STOP](#)" on page 615
- "[:POWer:CLResponse:YMAXimum](#)" on page 617
- "[:POWer:CLResponse:YMINimum](#)" on page 618

## :POWer:CLResponse:YMAXimum

**N** (see [page 1334](#))

**Command Syntax** :POWer:CLResponse:YMAXimum <value>

<value> ::= dB value in multiples of 10 from -110 to 120

The :POWer:CLResponse:YMAXimum command specifies the Bode plot's initial vertical scale maximum value.

This setting determines the initial scale and offset used in the Bode plot when the :POWer:CLResponse:APPLy command is sent (to start the analysis). When the analysis is complete, math function scale and offset commands can be used to adjust the settings.

**Query Syntax** :POWer:CLResponse:YMAXimum?

The :POWer:CLResponse:YMAXimum? query returns the vertical scale maximum setting.

**Return Format** <value><NL>

<value> ::= dB value in multiples of 10 from -110 to 120

- See Also**
- "[":POWer:CLResponse:APPLy](#)" on page 613
  - "[":POWer:CLResponse:FREQuency:STARt](#)" on page 614
  - "[":POWer:CLResponse:FREQuency:STOP](#)" on page 615
  - "[":POWer:CLResponse:VIEW](#)" on page 616
  - "[":POWer:CLResponse:YMINimum](#)" on page 618

## :POWer:CLResponse:YMINimum

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:CLResponse:YMINimum <value>`

`<value>` ::= dB value in multiples of 10 from -120 to 110

The :POWer:CLResponse:YMINimum command specifies the Bode plot's initial vertical scale minimum value.

This setting determines the initial scale and offset used in the Bode plot when the :POWer:CLResponse:APPLy command is sent (to start the analysis). When the analysis is complete, math function scale and offset commands can be used to adjust the settings.

**Query Syntax**    `:POWer:CLResponse:YMINimum?`

The :POWer:CLResponse:YMINimum? query returns the vertical scale minimum setting.

**Return Format**    `<value><NL>`

`<value>` ::= dB value in multiples of 10 from -120 to 110

**See Also**

- "[":POWer:CLResponse:APPLy](#)" on page 613
- "[":POWer:CLResponse:FREQuency:STARt](#)" on page 614
- "[":POWer:CLResponse:FREQuency:STOP](#)" on page 615
- "[":POWer:CLResponse:VIEW](#)" on page 616
- "[":POWer:CLResponse:YMAXimum](#)" on page 617

## :POWer:DESKew

**N** (see [page 1334](#))

**Command Syntax** :POWer:DESKew

The :POWer:DESKew command launches the auto deskew process on the oscilloscope.

Before sending this command:

- 1 Demagnetize and zero-adjust the current probe.  
Refer to the current probe's documentation for instructions on how to do this.
- 2 Make connections to the U1880A deskew fixture as described in the oscilloscope's connection dialog or in the *DSOX4PWR Power Measurement Application User's Guide*.
- 3 Make sure the voltage probe and current probe channels are specified appropriately using the :POWer:SIGNals:SOURce:VOLTage1 and :POWer:SIGNals:SOURce:CURRent1 commands.

### NOTE

Use the lowest attenuation setting on the high voltage differential probes whenever possible because the voltage levels on the deskew fixture are very small. Using a higher attenuation setting may yield inaccurate skew values (and affect the measurements made) because the noise level is magnified as well.

The deskew values are saved in the oscilloscope until a factory default or secure erase is performed. The next time you run the Power Application, you can use the saved deskew values or perform the deskew again.

Generally, you need to perform the deskew again when part of the test setup changes (for example, a different probe, different oscilloscope channel, etc.) or when the ambient temperature has changed.

**See Also**

- "[":POWer:SIGNals:SOURce:VOLTage<i>](#)" on page 669
- "[":POWer:SIGNals:SOURce:CURRent<i>](#)" on page 668

## :POWER:EFFiciency:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWER:EFFiciency:APPLy

The :POWER:EFFiciency:APPLy command applies the efficiency power analysis.

Efficiency analysis tests the overall efficiency of the power supply by measuring the output power over the input power.

**NOTE**

Efficiency analysis requires a 4-channel oscilloscope because input voltage, input current, output voltage, and output current are measured.

**See Also**

- "[:POWER:EFFiciency:TYPE](#)" on page 621
- "[:MEASure:EFFiciency](#)" on page 551
- "[:MEASure:IPOWer](#)" on page 554
- "[:MEASure:OPOWer](#)" on page 557

## :POWer:EFFiciency:TYPE

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:EFFiciency:TYPE <type>`  
`<type> ::= {DCDC | DCAC | ACDC | ACAC}`

The :POWer:EFFiciency:TYPE command specifies the type of power that is being converted from the input to the output. This selection affects how the efficiency is measured.

**Query Syntax**    `:POWer:EFFiciency:TYPE?`

The :POWer:EFFiciency:TYPE? query returns the currently specified type setting.

**Return Format**    `<type><NL>`  
`<type> ::= {DCDC | DCAC | ACDC | ACAC}`

**See Also**

- [":POWer:EFFiciency:APPLy"](#) on page 620
- [":MEASure:EFFiciency"](#) on page 551
- [":MEASure:IPOWer"](#) on page 554
- [":MEASure:OPOWer"](#) on page 557

## :POWer:ENABle

**N** (see [page 1334](#))

**Command Syntax** :POWer:ENABle {{0 | OFF} | {1 | ON}}

The :POWer:ENABle command enables or disables power analysis.

**Query Syntax** :POWer:ENABLe?

The :POWer:ENABLe query returns a 1 or a 0 showing whether power analysis is enabled or disabled, respectively.

**Return Format** {0 | 1}

**See Also** • [Chapter 23](#), “:MEASure Power Commands,” starting on page 543

## :POWer:HARMonics:APPLy



(see [page 1334](#))

### Command Syntax

`:POWer:HARMonics:APPLy`

The :POWer:HARMonics:APPLy command applies the current harmonics analysis.

Switching power supplies draw a range of harmonics from the AC mains.

Standard limits are set for these harmonics because these harmonics can travel back to the supply grid and cause problems with other devices on the grid.

Use the Current Harmonics analysis to test a switching power supply's current harmonics to pre-compliance standard of IEC61000-3-2 (Class A, B, C, or D). The analysis presents up to 40 harmonics.

### See Also

- [":POWer:HARMonics:DATA"](#) on page 624
- [":POWer:HARMonics:DISPlay"](#) on page 625
- [":POWer:HARMonics:FAILcount"](#) on page 626
- [":POWer:HARMonics:LINE"](#) on page 627
- [":POWer:HARMonics:POWERfactor"](#) on page 628
- [":POWer:HARMonics:STANDARD"](#) on page 632
- [":POWer:HARMonics:STATUS"](#) on page 633
- [":POWer:HARMonics:RUNCount"](#) on page 631
- [":POWer:HARMonics:THD"](#) on page 634

**:POWER:HARMonics:DATA**(see [page 1334](#))**Query Syntax**    `:POWER:HARMonics:DATA?`

The `:POWER:HARMonics:DATA` query returns the power harmonics results table data.

**Return Format**    `<binary_block> ::= comma-separated data with newlines at the end of each row`

- See Also**
- [":POWER:HARMonics:APPLy"](#) on page 623
  - [":POWER:HARMonics:DISPlay"](#) on page 625
  - [":POWER:HARMonics:FAILcount"](#) on page 626
  - [":POWER:HARMonics:LINE"](#) on page 627
  - [":POWER:HARMonics:POWERfactor"](#) on page 628
  - [":POWER:HARMonics:RUNCount"](#) on page 631
  - [":POWER:HARMonics:STANDARD"](#) on page 632
  - [":POWER:HARMonics:STATUS"](#) on page 633
  - [":POWER:HARMonics:THD"](#) on page 634

## :POWer:HARMonics:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:HARMonics:DISPlay <display>`  
`<display> ::= {TABLe | BAR | OFF}`

The :POWer:HARMonics:DISPlay command specifies how to display the current harmonics analysis results:

- TABLe
- BAR – Bar chart.
- OFF – Harmonics measurement results are not displayed.

**Query Syntax**    `:POWer:HARMonics:DISPlay?`

The :POWer:HARMonics:DISPlay query returns the display setting.

**Return Format**    `<display><NL>`  
`<display> ::= {TABL | BAR | OFF}`

**See Also**    [":POWer:HARMonics:APPLy" on page 623](#)  
[":POWer:HARMonics:DATA" on page 624](#)  
[":POWer:HARMonics:FAILcount" on page 626](#)  
[":POWer:HARMonics:LINE" on page 627](#)  
[":POWer:HARMonics:POWERfactor" on page 628](#)  
[":POWer:HARMonics:RUNCount" on page 631](#)  
[":POWer:HARMonics:STANDARD" on page 632](#)  
[":POWer:HARMonics:STATUS" on page 633](#)  
[":POWer:HARMonics:THD" on page 634](#)

**:POWer:HARMonics:FAILcount**

**N** (see [page 1334](#))

**Query Syntax** `:POWer:HARMonics:FAILcount?`

Returns the current harmonics analysis' fail count. Non Spec values (that is, harmonics values not specified by the selected standard) are not counted.

**Return Format** `<count><NL>`

`<count>` ::= integer in NR1 format

- See Also**
- "[":POWer:HARMonics:RUNCount](#)" on page 631
  - "[":POWer:HARMonics:APPLy](#)" on page 623
  - "[":POWer:HARMonics:DATA](#)" on page 624
  - "[":POWer:HARMonics:DISPlay](#)" on page 625
  - "[":POWer:HARMonics:LINE](#)" on page 627
  - "[":POWer:HARMonics:POWERfactor](#)" on page 628
  - "[":POWer:HARMonics:STANDARD](#)" on page 632
  - "[":POWer:HARMonics:STATUS](#)" on page 633
  - "[":POWer:HARMonics:THD](#)" on page 634

## :POWER:HARMonics:LINE

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:HARMonics:LINE <frequency>`  
`<frequency> ::= {F50 | F60 | F400}`

The :POWER:HARMonics:LINE command specifies the line frequency setting for the current harmonics analysis:

- F50 – 50 Hz.
- F60 – 60 Hz.
- F400 – 400 Hz.

**Query Syntax**    `:POWER:HARMonics:LINE?`

The :POWER:HARMonics:LINE query returns the line frequency setting.

**Return Format**    `<frequency><NL>`  
`<frequency> ::= {F50 | F60 | F400}`

**See Also**    [":POWER:HARMonics:APPLy" on page 623](#)  
[":POWER:HARMonics:DATA" on page 624](#)  
[":POWER:HARMonics:DISPlay" on page 625](#)  
[":POWER:HARMonics:FAILcount" on page 626](#)  
[":POWER:HARMonics:POWERfactor" on page 628](#)  
[":POWER:HARMonics:RUNCount" on page 631](#)  
[":POWER:HARMonics:STANDARD" on page 632](#)  
[":POWER:HARMonics:STATUS" on page 633](#)  
[":POWER:HARMonics:THD" on page 634](#)

**:POWer:HARMonics:POWerfactor**(see [page 1334](#))**Query Syntax**    `:POWer:HARMonics:POWerfactor?`

The `:POWer:HARMonics:POWerfactor` query returns the power factor for IEC 61000-3-2 Standard Class C power factor value.

**Return Format**    `<value> ::= Class C power factor in NR3 format`**See Also**

- "[":POWer:HARMonics:APPLy](#)" on page 623
- "[":POWer:HARMonics:DATA](#)" on page 624
- "[":POWer:HARMonics:DISPlay](#)" on page 625
- "[":POWer:HARMonics:FAILcount](#)" on page 626
- "[":POWer:HARMonics:LINE](#)" on page 627
- "[":POWer:HARMonics:RUNCount](#)" on page 631
- "[":POWer:HARMonics:STANDARD](#)" on page 632
- "[":POWer:HARMonics:STATUS](#)" on page 633
- "[":POWer:HARMonics:THD](#)" on page 634

## :POWer:HARMonics:RPOWer

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:HARMonics:RPOWer <source>`  
`<source> ::= {MEASured | USER}`

When Class D is selected as the current harmonics analysis standard, the :POWer:HARMonics:RPOWer command specifies whether the Real Power value used for the current-per-watt measurement is measured by the oscilloscope or is defined by the user.

When USER is selected, use the :POWer:HARMonics:RPOWer:USER command to enter the user-defined value.

**Query Syntax**    `:POWer:HARMonics:RPOWer?`

The :POWer:HARMonics:RPOWer? query returns the Real Power source setting.

**Return Format**    `<source><NL>`  
`<source> ::= {MEAS | USER}`

**See Also**

- "[":POWer:HARMonics:STANDARD](#)" on page 632
- "[":POWer:HARMonics:RPOWer:USER](#)" on page 630

**:POWer:HARMonics:RPOWer:USER**

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:HARMonics:RPOWer:USER <value>`

`<value> ::= Watts from 1.0 to 600.0 in NR3 format`

When Class D is selected as the current harmonics analysis standard and you have chosen to use a user-defined Real Power value (see :POWer:HARMonics:RPOWer), the :POWer:HARMonics:RPOWer:USER command specifies the Real Power value used in the current-per-watt measurement.

**Query Syntax**    `:POWer:HARMonics:RPOWer:USER?`

The :POWer:HARMonics:RPOWer:USER? query returns the user-defined Real Power value.

**Return Format**    `<value><NL>`

`<value> ::= Watts from 1.0 to 600.0 in NR3 format`

**See Also**

- [":POWer:HARMonics:STANdard"](#) on page 632
- [":POWer:HARMonics:RPOWer"](#) on page 629

**:POWer:HARMonics:RUNCount**

**N** (see [page 1334](#))

**Query Syntax** :POWer:HARMonics:RUNCount?

Returns the current harmonics analysis' run iteration count. Non Spec values (that is, harmonics values not specified by the selected standard) are not counted.

**Return Format** <count><NL>

<count> ::= integer in NR1 format

**See Also**

- "[:POWer:HARMonics:FAILcount](#)" on page 626
- "[:POWer:HARMonics:APPLy](#)" on page 623
- "[:POWer:HARMonics:DATA](#)" on page 624
- "[:POWer:HARMonics:DISPLAY](#)" on page 625
- "[:POWer:HARMonics:LINE](#)" on page 627
- "[:POWer:HARMonics:POWERfactor](#)" on page 628
- "[:POWer:HARMonics:STANDARD](#)" on page 632
- "[:POWer:HARMonics:STATUS](#)" on page 633
- "[:POWer:HARMonics:THD](#)" on page 634

## :POWER:HARMonics:STANDARD

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:HARMonics:STANDARD <class>`  
`<class> ::= {A | B | C | D}`

The :POWER:HARMonics:STANDARD command selects the standard to perform current harmonics compliance testing on.

- A – IEC 61000-3-2 Class A – for balanced three-phase equipment, household appliances (except equipment identified as Class D), tools excluding portable tools, dimmers for incandescent lamps, and audio equipment.
- B – IEC 61000-3-2 Class B – for portable tools.
- C – IEC 61000-3-2 Class C – for lighting equipment.
- D – IEC 61000-3-2 Class D – for equipment having a specified power according less than or equal to 600 W, of the following types: personal computers and personal computer monitors, television receivers.

**Query Syntax**    `:POWER:HARMonics:STANDARD?`

The :POWER:HARMonics:STANDARD query returns the currently set IEC 61000-3-2 standard.

**Return Format**    `<class><NL>`  
`<class> ::= {A | B | C | D}`

**See Also**    [":POWER:HARMonics:APPLy"](#) on page 623  
[":POWER:HARMonics:DATA"](#) on page 624  
[":POWER:HARMonics:DISPLAY"](#) on page 625  
[":POWER:HARMonics:FAILcount"](#) on page 626  
[":POWER:HARMonics:LINE"](#) on page 627  
[":POWER:HARMonics:POWERfactor"](#) on page 628  
[":POWER:HARMonics:RUNCount"](#) on page 631  
[":POWER:HARMonics:STATus"](#) on page 633  
[":POWER:HARMonics:THD"](#) on page 634

**:POWer:HARMonics:STATUs****N** (see [page 1334](#))**Query Syntax** `:POWer:HARMonics:STATUs?`

The :POWer:HARMonics:STATUs query returns the overall pass/fail status of the current harmonics analysis.

**Return Format** `<status> ::= {PASS | FAIL | UNTested}`**See Also**

- "[:POWer:HARMonics:RUNCount](#)" on page 631
- "[:POWer:HARMonics:FAILcount](#)" on page 626
- "[:POWer:HARMonics:APPLy](#)" on page 623
- "[:POWer:HARMonics:DATA](#)" on page 624
- "[:POWer:HARMonics:DISPLAY](#)" on page 625
- "[:POWer:HARMonics:LINE](#)" on page 627
- "[:POWer:HARMonics:POWERfactor](#)" on page 628
- "[:POWer:HARMonics:STANDARD](#)" on page 632
- "[:POWer:HARMonics:THD](#)" on page 634

## :POWer:HARMonics:THD

**N** (see [page 1334](#))

**Query Syntax** `:POWer:HARMonics:THD?`

The `:POWer:HARMonics:THD` query returns the Total Harmonics Distortion (THD) results of the current harmonics analysis.

**Return Format** `<value> ::= Total Harmonics Distortion in NR3 format`

**See Also**

- "[":POWer:HARMonics:APPLy](#)" on page 623
- "[":POWer:HARMonics:DATA](#)" on page 624
- "[":POWer:HARMonics:DISPlay](#)" on page 625
- "[":POWer:HARMonics:FAILcount](#)" on page 626
- "[":POWer:HARMonics:LINE](#)" on page 627
- "[":POWer:HARMonics:POWERfactor](#)" on page 628
- "[":POWer:HARMonics:RUNCount](#)" on page 631
- "[":POWer:HARMonics:STANDARD](#)" on page 632
- "[":POWer:HARMonics:STATUS](#)" on page 633

## :POWer:INRush:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:INRush:APPLy

The :POWer:INRush:APPLy command applies the inrush current analysis.

The Inrush current analysis measures the peak inrush current of the power supply when the power supply is first turned on.

- See Also**
- "[:MEASure:PCURrent](#)" on page 558
  - "[:POWer:INRush:EXIT](#)" on page 636
  - "[:POWer:INRush:NEXT](#)" on page 637

## :POWer:INRush:EXIT

**N** (see [page 1334](#))

**Command Syntax** :POWer:INRush:EXIT

The :POWer:INRush:EXIT command exits (stops) the inrush current power analysis.

This command is equivalent to pressing the **Exit** softkey on the oscilloscope front panel during the analysis.

**See Also**

- "[:POWer:INRush:APPLy](#)" on page 635
- "[:POWer:INRush:NEXT](#)" on page 637

## :POWer:INRush:NEXT

**N** (see [page 1334](#))

**Command Syntax** :POWer:INRush:NEXT

The :POWer:INRush:NEXT command goes to the next step of the inrush current analysis.

This command is equivalent to pressing the **Next** softkey on the oscilloscope front panel when prompted during the analysis.

**See Also**

- "[:POWer:INRush:APPLy](#)" on page 635
- "[:POWer:INRush:EXIT](#)" on page 636

## :POWer:MODulation:APPLy



(see [page 1334](#))

**Command Syntax** :POWer:MODulation:APPLy

The :POWer:MODulation:APPLy command applies the selected modulation analysis type (:POWer:MODulation:TYPE).

The Modulation analysis measures the control pulse signal to a switching device (MOSFET) and observes the trending of the pulse width, duty cycle, period, frequency, etc. of the control pulse signal.

**See Also**

- [":POWer:MODulation:SOURce"](#) on page 639
- [":POWer:MODulation:TYPE"](#) on page 640
- [":MEASure:VAVerage"](#) on page 530
- [":MEASure:VRMS"](#) on page 536
- [":MEASure:VRATio"](#) on page 535
- [":MEASure:PERiod"](#) on page 506
- [":MEASure:FREQuency"](#) on page 498
- [":MEASure:PWidth"](#) on page 510
- [":MEASure:NWIDth"](#) on page 502
- [":MEASure:DUTYcycle"](#) on page 496
- [":MEASure:RISetime"](#) on page 514
- [":MEASure:FALLtime"](#) on page 497

**:POWer:MODulation:SOURce**

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:MODulation:SOURce <source>`  
                  `<source> ::= {V | I}`

The :POWer:MODulation:SOURce command selects either the voltage source or the current source as the source for the modulation analysis.

**Query Syntax**    `:POWer:MODulation:SOURce?`

The :POWer:MODulation:SOURce query returns the selected source for the modulation analysis.

**Return Format**    `<source><NL>`  
                  `<source> ::= {V | I}`

**See Also**

- [":POWer:MODulation:APPLy" on page 638](#)
- [":POWer:MODulation:TYPE" on page 640](#)

## :POWER:MODulation:TYPE

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:POWER:MODulation:TYPE &lt;modulation&gt;</code>
	$<\text{modulation}> ::= \{\text{VAVerage} \mid \text{ACRMs} \mid \text{VRATio} \mid \text{PERiod} \mid \text{FREQuency} \mid \text{PWIDith} \mid \text{NWIDth} \mid \text{DUTYcycle} \mid \text{RISetime} \mid \text{FALLtime}\}$

The :POWER:MODulation:TYPE command selects the type of measurement to make in the modulation analysis:

- VAVerage
- ACRMs
- VRATio
- PERiod
- FREQuency
- PWIDth (positive pulse width)
- NWIDth (negative pulse width)
- DUTYcycle
- RISetime
- FALLtime

<b>Query Syntax</b>	<code>:POWER:MODulation:TYPE?</code>
---------------------	--------------------------------------

The :POWER:MODulation:TYPE query returns the modulation type setting.

<b>Return Format</b>	<code>&lt;modulation&gt;&lt;NL&gt;</code>
	$<\text{modulation}> ::= \{\text{VAV} \mid \text{ACRM} \mid \text{VRAT} \mid \text{PER} \mid \text{FREQ} \mid \text{PWID} \mid \text{NWID} \mid \text{DUTY} \mid \text{RIS} \mid \text{FALL}\}$

<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">:POWER:MODulation:SOURce</a>" on page 639</li> <li>• "<a href="#">:POWER:MODulation:APPLy</a>" on page 638</li> <li>• "<a href="#">:MEASure:VAVerage</a>" on page 530</li> <li>• "<a href="#">:MEASure:VRMS</a>" on page 536</li> <li>• "<a href="#">:MEASure:VRATio</a>" on page 535</li> <li>• "<a href="#">:MEASure:PERiod</a>" on page 506</li> <li>• "<a href="#">:MEASure:FREQuency</a>" on page 498</li> <li>• "<a href="#">:MEASure:PWIDth</a>" on page 510</li> <li>• "<a href="#">:MEASure:NWIDth</a>" on page 502</li> <li>• "<a href="#">:MEASure:DUTYcycle</a>" on page 496</li> <li>• "<a href="#">:MEASure:RISetime</a>" on page 514</li> <li>• "<a href="#">:MEASure:FALLtime</a>" on page 497</li> </ul>
-----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :POWer:ONOOff:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:ONOOff:APPLy

The :POWer:ONOOff:APPLy command applies the selected turn on/off analysis test (:POWer:ONOOff:TEST).

**See Also**

- "[:POWer:SIGNals:VSteady:ONOOff:OFF](#)" on page 665
- "[:POWer:SIGNals:VSteady:ONOOff:ON](#)" on page 666
- "[:MEASure:ONTIme](#)" on page 556
- "[:MEASure:OFFTime](#)" on page 555
- "[:POWer:ONOOff:TEST](#)" on page 644
- "[:POWer:ONOOff:EXIT](#)" on page 642
- "[:POWer:ONOOff:NEXT](#)" on page 643

## :POWer:ONOOff:EXIT

**N** (see [page 1334](#))

**Command Syntax** :POWer:ONOOff:EXIT

The :POWer:ONOOff:EXIT command exits (stops) the turn on time/turn off time analysis.

This command is equivalent to pressing the **Exit** softkey on the oscilloscope front panel during the analysis.

- See Also**
- "[":POWer:ONOOff:APPLy](#)" on page 641
  - "[":POWer:ONOOff:NEXT](#)" on page 643
  - "[":POWer:ONOOff:TEST](#)" on page 644

## :POWer:ONOOff:NEXT

**N** (see [page 1334](#))

**Command Syntax** :POWer:ONOOff:NEXT

The :POWer:ONOOff:NEXT command goes to the next step of the turn on/turn off analysis.

This command is equivalent to pressing the **Next** softkey on the oscilloscope front panel when prompted during the analysis.

- See Also**
- "[:POWer:ONOOff:APPLy](#)" on page 641
  - "[:POWer:ONOOff:EXIT](#)" on page 642
  - "[:POWer:ONOOff:TEST](#)" on page 644

**:POWER:ONOFF:TEST****N** (see [page 1334](#))**Command Syntax**    `:POWER:ONOFF:TEST {{0 | OFF} | {1 | ON}}`

The :POWER:ONOFF:TEST command selects whether turn on or turn off analysis is performed:

- ON – Turn On – measures the time taken to get the output voltage of the power supply after the input voltage is applied.
- OFF – Turn Off – measures the time taken for the output voltage of the power supply to turn off after the input voltage is removed.

**Query Syntax**    `:POWER:ONOFF:TEST?`

The :POWER:ONOFF:TEST query returns the selected test type.

**Return Format**    `{0 | 1}`**See Also**

- "[":POWER:ONOFF:APPLy](#)" on page 641
- "[":POWER:ONOFF:EXIT](#)" on page 642
- "[":POWER:ONOFF:NEXT](#)" on page 643

## :POWer:PSRR:APPLy

**N** (see [page 1334](#))

### Command Syntax

`:POWer:PSRR:APPLy`

The :POWer:PSRR:APPLy command applies the power supply rejection ratio (PSRR) analysis.

The Power Supply Rejection Ratio (PSRR) test is used to determine how well a voltage regulator rejects ripple noise over different frequency range.

This analysis provides a signal from the oscilloscope's waveform generator that sweeps its frequency. This signal is used to inject ripple to the DC voltage that feeds the voltage regulator.

The AC RMS ratio of the input over the output is measured and is plotted over the range of frequencies.

It takes some time for the frequency sweep analysis to complete. You can query bit 0 of the Standard Event Status Register (\*ESR?) to find out when the analysis is complete.

### See Also

- [":POWer:PSRR:FREQuency:MAXimum" on page 646](#)
- [":POWer:PSRR:FREQuency:MINimum" on page 647](#)
- [":POWer:PSRR:RMAXimum" on page 648](#)
- ["\\*ESR \(Standard Event Status Register\)" on page 184](#)

## :POWER:PSRR:FREQuency:MAXimum

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:PSRR:FREQuency:MAXimum <value>[suffix]`

`<value> ::= {10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000`  
`| 20000000}`

`[suffix] ::= {Hz | kHz | MHz}`

The :POWER:PSRR:FREQuency:MAXimum command sets the end sweep frequency value. The PSRR measurement is displayed on a log scale, so you can select from decade values in addition to the maximum frequency of 20 MHz.

**Query Syntax**    `:POWER:PSRR:FREQuency:MAXimum?`

The :POWER:PSRR:FREQuency:MAXimum query returns the maximum sweep frequency setting.

**Return Format**    `<value><NL>`

`<value> ::= {10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000`  
`| 20000000}`

**See Also**

- "[":POWER:PSRR:APPLY](#)" on page 645
- "[":POWER:PSRR:FREQuency:MINimum](#)" on page 647
- "[":POWER:PSRR:RMAXimum](#)" on page 648

## :POWer:PSRR:FREQuency:MINimum

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:PSRR:FREQuency:MINimum <value>[suffix]`

`<value> ::= {1 | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}`

`[suffix] ::= {Hz | kHz | MHz}`

The :POWer:PSRR:FREQuency:MINimum command sets the start sweep frequency value. The measurement is displayed on a log scale, so you can select from decade values.

**Query Syntax**    `:POWer:PSRR:FREQuency:MINimum?`

The :POWer:PSRR:FREQuency:MINimum query returns the minimum sweep frequency setting.

**Return Format**    `<value><NL>`

`<value> ::= {1 | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}`

**See Also**

- [":POWer:PSRR:APPLy"](#) on page 645
- [":POWer:PSRR:FREQuency:MAXimum"](#) on page 646
- [":POWer:PSRR:RMAXimum"](#) on page 648

**:POWer:PSRR:RMAXimum****N** (see [page 1334](#))**Command Syntax**    `:POWer:PSRR:RMAXimum <value>``<value> ::= Maximum ratio value in NR1 format`

The :POWer:PSRR:RMAXimum command specifies the vertical scale of the PSRR math waveform.

This setting determines the initial scale and offset used in the PSRR plot when the :POWer:PSRR:APPLy command is sent (to start the analysis). When the analysis is complete, math function scale and offset commands can be used to adjust the settings.

**Query Syntax**    `:POWer:PSRR:RMAXimum?`

The :POWer:PSRR:RMAXimum query returns the currently specified maximum ratio setting.

**Return Format**    `<value><NL>``<value> ::= Maximum ratio value in NR1 format`**See Also**

- "[":POWer:PSRR:APPLy](#)" on page 645
- "[":POWer:PSRR:RMAXimum](#)" on page 648
- "[":POWer:PSRR:FREQuency:MAXimum](#)" on page 646
- "[":POWer:PSRR:FREQuency:MINimum](#)" on page 647

## :POWer:QUALity:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:QUALity:APPLy

The :POWer:QUALity:APPLy command applies the selected power quality analysis type (:POWer:QUALity:TYPE).

The power quality analysis shows the quality of the AC input line.

Some AC current may flow back into and back out of the load without delivering energy. This current, called reactive or harmonic current, gives rise to an "apparent" power which is larger than the actual power consumed. Power quality is gauged by these measurements: power factor, apparent power, true power, reactive power, crest factor, and phase angle of the current and voltage of the AC line.

- See Also**
- "[:MEASure:FACTOr](#)" on page 553
  - "[:MEASure:REAL](#)" on page 562
  - "[:MEASure:APPARENT](#)" on page 548
  - "[:MEASure:REACTIVE](#)" on page 561
  - "[:MEASure:CREST](#)" on page 550
  - "[:MEASure:ANGLE](#)" on page 547

## :POWer:RIPPLe:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:RIPPLe:APPLy

The :POWer:RIPPLe:APPLy command applies the output ripple analysis.

**See Also** • [":MEASure:RIPPLe"](#) on page 563

## :POWer:SIGNals:AUTosetup

**N** (see [page 1334](#))

**Command Syntax** :POWer:SIGNals:AUTosetup <analysis>

```
<analysis> ::= {HARMonics | EFFiciency | RIPPLE | MODulation | QUALity
| SLEW | SWITch | RDSVce}
```

The :POWer:SIGNals:AUTosetup command performs automated oscilloscope setup for the signals in the specified type of power analysis.

**See Also**

- "[":POWer:HARMonics:DISPLAY](#)" on page 625
- "[":POWer:EFFiciency:APPLy](#)" on page 620
- "[":POWer:RIPPLE:APPLy](#)" on page 650
- "[":POWer:MODulation:APPLy](#)" on page 638
- "[":POWer:QUALity:APPLy](#)" on page 649
- "[":POWer:SLEW:APPLy](#)" on page 670
- "[":POWer:SWITch:APPLy](#)" on page 672
- "[":POWer:SIGNals:CYCLES:HARMonics](#)" on page 652
- "[":POWer:SIGNals:CYCLES:QUALity](#)" on page 653
- "[":POWer:SIGNals:DURation:EFFiciency](#)" on page 654
- "[":POWer:SIGNals:DURation:MODulation](#)" on page 655
- "[":POWer:SIGNals:DURation:RIPPLE](#)" on page 658
- "[":POWer:SIGNals:IEXPected](#)" on page 660
- "[":POWer:SIGNals:OVERshoot](#)" on page 661
- "[":POWer:SIGNals:SOURce:CURREnt<i>](#)" on page 668
- "[":POWer:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:CYCLeS:HARMonics

**N** (see [page 1334](#))

**Command Syntax**

```
:POWER:SIGNals:CYCLeS:HARMonics <count>
<count> ::= integer in NR1 format
Legal values are 1 to 100.
```

The :POWER:SIGNals:CYCLeS:HARMonics command specifies the number of cycles to include in the current harmonics analysis.

**Query Syntax**

```
:POWER:SIGNals:CYCLeS:HARMonics?
```

The :POWER:SIGNals:CYCLeS:HARMonics query returns the number of cycles currently set.

**Return Format**

```
<count><NL>
<count> ::= integer in NR1 format
```

**See Also**

- "[":POWER:HARMonics:DISPlay](#)" on page 625
- "[":POWER:HARMonics:APPLy](#)" on page 623
- "[":POWER:SIGNals:AUTosetup](#)" on page 651
- "[":POWER:SIGNals:IEXPected](#)" on page 660
- "[":POWER:SIGNals:OVERshoot](#)" on page 661
- "[":POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
- "[":POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWer:SIGNals:CYCLes:QUALity

**N** (see [page 1334](#))

**Command Syntax**

```
:POWer:SIGNals:CYCLes:QUALity <count>
<count> ::= integer in NR1 format
Legal values are 1 to 100.
```

The :POWer:SIGNals:CYCLes:QUALity command specifies the number of cycles to include in the power quality analysis.

**Query Syntax**

```
:POWer:SIGNals:CYCLes:QUALity?
```

The :POWer:SIGNals:CYCLes:QUALity query returns the number of cycles currently set.

**Return Format**

```
<count><NL>
```

<count> ::= integer in NR1 format

**See Also**

- "[:POWer:QUALity:APPLy](#)" on page 649
- "[:POWer:SIGNals:AUTosetup](#)" on page 651
- "[:POWer:SIGNals:IEXPected](#)" on page 660
- "[:POWer:SIGNals:OVERshoot](#)" on page 661
- "[:POWer:SIGNals:SOURce:CURRent<i>](#)" on page 668
- "[:POWer:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:DURation:EFFiciency

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:DURation:EFFiciency <value>[suffix]`

`<value>` ::= value in NR3 format

`[suffix]` ::= {s | ms | us | ns}

The :POWER:SIGNals:DURation:EFFiciency command specifies the duration of the efficiency analysis.

**Query Syntax**    `:POWER:SIGNals:DURation:EFFiciency?`

The :POWER:SIGNals:DURation:EFFiciency query returns the set duration time value.

**Return Format**    `<value><NL>`

`<value>` ::= value in NR3 format

- See Also**
- "[":POWER:EFFiciency:APPLy](#)" on page 620
  - "[":POWER:SIGNals:AUTosetup](#)" on page 651
  - "[":POWER:SIGNals:IEXPected](#)" on page 660
  - "[":POWER:SIGNals:OVERshoot](#)" on page 661
  - "[":POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
  - "[":POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWer:SIGNals:DURation:MODulation

**N** (see [page 1334](#))

**Command Syntax** :POWer:SIGNals:DURation:MODulation <value>[suffix]

<value> ::= value in NR3 format

[suffix] ::= {s | ms | us | ns}

The :POWer:SIGNals:DURation:MODulation command specifies the duration of the modulation analysis.

**Query Syntax** :POWer:SIGNals:DURation:MODulation?

The :POWer:SIGNals:DURation:MODulation query returns the set duration time value.

**Return Format** <value><NL>

<value> ::= value in NR3 format

- See Also**
- "[:POWer:MODulation:APPLy](#)" on page 638
  - "[:POWer:SIGNals:AUTosetup](#)" on page 651
  - "[:POWer:SIGNals:IEXPected](#)" on page 660
  - "[:POWer:SIGNals:OVERshoot](#)" on page 661
  - "[:POWer:SIGNals:SOURce:CURRent<i>](#)" on page 668
  - "[:POWer:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:DURation:ONOFF:OFF

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:DURation:ONOFF:OFF <value>[suffix]`

`<value>` ::= value in NR3 format

`[suffix]` ::= {s | ms | us | ns}

The :POWER:SIGNals:DURation:ONOFF:OFF command specifies the duration of the turn off analysis.

**Query Syntax**    `:POWER:SIGNals:DURation:ONOFF:OFF?`

The :POWER:SIGNals:DURation:ONOFF:OFF query returns the set duration time value.

**Return Format**    `<value><NL>`

`<value>` ::= value in NR3 format

**See Also**

- [":POWER:ONOFF:APPLy" on page 641](#)
- [":POWER:SIGNals:AUTosetup" on page 651](#)
- [":POWER:SIGNals:IEXPected" on page 660](#)
- [":POWER:SIGNals:OVERshoot" on page 661](#)
- [":POWER:SIGNals:VMAXimum:ONOFF:OFF" on page 663](#)
- [":POWER:SIGNals:VSTeady:ONOFF:OFF" on page 665](#)
- [":POWER:SIGNals:SOURce:CURREnt<i>" on page 668](#)
- [":POWER:SIGNals:SOURce:VOLTage<i>" on page 669](#)

## :POWer:SIGNals:DURation:ONOFF:ON

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:SIGNals:DURation:ONOFF:ON <value>[suffix]`

`<value>` ::= value in NR3 format

`[suffix]` ::= {s | ms | us | ns}

The :POWer:SIGNals:DURation:ONOFF:ON command specifies the duration of the turn on analysis.

**Query Syntax**    `:POWer:SIGNals:DURation:ONOFF:ON?`

The :POWer:SIGNals:DURation:ONOFF:ON query returns the set duration time value.

**Return Format**    `<value><NL>`

`<value>` ::= value in NR3 format

**See Also**

- [":POWer:ONOFF:APPLy" on page 641](#)
- [":POWer:SIGNals:AUTosetup" on page 651](#)
- [":POWer:SIGNals:IEXPected" on page 660](#)
- [":POWer:SIGNals:OVERshoot" on page 661](#)
- [":POWer:SIGNals:VMAXimum:ONOFF:ON" on page 664](#)
- [":POWer:SIGNals:VSTeady:ONOFF:ON" on page 666](#)
- [":POWer:SIGNals:SOURce:CURREnt<i>" on page 668](#)
- [":POWer:SIGNals:SOURce:VOLTage<i>" on page 669](#)

## :POWER:SIGNals:DURation:RIPPle

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:DURation:RIPPle <value>[suffix]`

`<value>` ::= value in NR3 format

`[suffix]` ::= {s | ms | us | ns}

The :POWER:SIGNals:DURation:RIPPle command specifies the duration of the output ripple analysis.

**Query Syntax**    `:POWER:SIGNals:DURation:RIPPle?`

The :POWER:SIGNals:DURation:RIPPle query returns the set duration time value.

**Return Format**    `<value><NL>`

`<value>` ::= value in NR3 format

- See Also**
- "[":POWER:RIPPLE:APPLY](#)" on page 650
  - "[":POWER:SIGNals:AUTosetup](#)" on page 651
  - "[":POWER:SIGNals:IEXPected](#)" on page 660
  - "[":POWER:SIGNals:OVERshoot](#)" on page 661
  - "[":POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
  - "[":POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWer:SIGNals:DURation:TRANsient

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:SIGNals:DURation:TRANsient <value>[suffix]`

`<value>` ::= value in NR3 format

`[suffix]` ::= {s | ms | us | ns}

The :POWer:SIGNals:DURation:TRANsient command specifies the duration of the transient response analysis.

**Query Syntax**    `:POWer:SIGNals:DURation:TRANsient?`

The :POWer:SIGNals:DURation:TRANsient query returns the set duration time value.

**Return Format**    `<value><NL>`

`<value>` ::= value in NR3 format

**See Also**

- [":POWer:TRANsient:APPLy" on page 678](#)
- [":POWer:SIGNals:AUTosetup" on page 651](#)
- [":POWer:SIGNals:IEXPected" on page 660](#)
- [":POWer:SIGNals:OVERshoot" on page 661](#)
- [":POWer:SIGNals:VSTeady:TRANsient" on page 667](#)
- [":POWer:SIGNals:SOURce:CURREnt<i>" on page 668](#)
- [":POWer:SIGNals:SOURce:VOLTage<i>" on page 669](#)

**:POWER:SIGNals:IEXPected****N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:IEXPected <value>[suffix]`  
`<value> ::= Expected current value in NR3 format`  
`[suffix] ::= {A | mA}`

The :POWER:SIGNals:IEXPected command specifies the expected inrush current amplitude. This value is used to set the vertical scale of the channel probing current.

**Query Syntax**    `:POWER:SIGNals:IEXPected?`

The :POWER:SIGNals:IEXPected query returns the expected inrush current setting.

**Return Format**    `<value><NL>`  
`<value> ::= Expected current value in NR3 format`

**See Also**

- [":POWER:INRush:APPLy" on page 635](#)
- [":POWER:SIGNals:AUTosetup" on page 651](#)
- [":POWER:SIGNals:OVERshoot" on page 661](#)
- [":POWER:SIGNals:VMAXimum:INRush" on page 662](#)
- [":POWER:SIGNals:SOURce:CURREnt<i>" on page 668](#)
- [":POWER:SIGNals:SOURce:VOLTage<i>" on page 669](#)

## :POWER:SIGNals:OVERshoot

**N** (see [page 1334](#))

**Command Syntax** :POWER:SIGNals:OVERshoot <percent>

<percent> ::= percent of overshoot value in NR1 format

The :POWER:SIGNals:OVERshoot command specifies the percent of overshoot of the output voltage. This value is used to determine the settling band value for the transient response and to adjust the vertical scale of the oscilloscope.

**Query Syntax** :POWER:SIGNals:OVERshoot?

The :POWER:SIGNals:OVERshoot query returns the overshoot percent setting.

**Return Format** <percent><NL>

<percent> ::= percent of overshoot value in NR1 format

**See Also**

- "[:POWER:TRANsient:APPLy](#)" on page 678
- "[:POWER:SIGNals:AUTosetup](#)" on page 651
- "[:POWER:SIGNals:DURation:TRANsient](#)" on page 659
- "[:POWER:SIGNals:IEXPected](#)" on page 660
- "[:POWER:SIGNals:VSTeady:TRANsient](#)" on page 667
- "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
- "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VMAXimum:INRush

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:VMAXimum:INRush <value>[suffix]`

`<value>` ::= Maximum expected input Voltage in NR3 format

`[suffix]` ::= {v | mV}

The :POWER:SIGNals:VMAXimum:INRush command specifies the maximum expected input voltage. This value is used to set the vertical scale of the channel probing voltage for inrush current analysis.

**Query Syntax**    `:POWER:SIGNals:VMAXimum:INRush?`

The :POWER:SIGNals:VMAXimum:INRush query returns the expected maximum input voltage setting.

**Return Format**    `<value><NL>`

`<value>` ::= Maximum expected input Voltage in NR3 format

- See Also**
- "[:POWER:INRush:APPLY](#)" on page 635
  - "[:POWER:SIGNals:AUTosetup](#)" on page 651
  - "[:POWER:SIGNals:IEXPected](#)" on page 660
  - "[:POWER:SIGNals:OVERshoot](#)" on page 661
  - "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
  - "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VMAXimum:ONOFF:OFF

**N** (see [page 1334](#))

**Command Syntax** :POWER:SIGNals:VMAXimum:ONOFF:OFF <value>[suffix]

<value> ::= Maximum expected input Voltage in NR3 format

[suffix] ::= {v | mV}

The :POWER:SIGNals:VMAXimum:ONOFF:OFF command specifies the maximum expected input voltage. This value is used to set the vertical scale of the channel probing voltage for turn off analysis.

**Query Syntax** :POWER:SIGNals:VMAXimum:ONOFF:OFF?

The :POWER:SIGNals:VMAXimum:ONOFF:OFF query returns the expected maximum input voltage setting.

**Return Format** <value><NL>

<value> ::= Maximum expected input Voltage in NR3 format

- See Also**
- "[:POWER:ONOFF:APPLy](#)" on page 641
  - "[:POWER:SIGNals:AUTosetup](#)" on page 651
  - "[:POWER:SIGNals:DURation:ONOFF:OFF](#)" on page 656
  - "[:POWER:SIGNals:IEXPected](#)" on page 660
  - "[:POWER:SIGNals:OVERshoot](#)" on page 661
  - "[:POWER:SIGNals:VSTeady:ONOFF:OFF](#)" on page 665
  - "[:POWER:SIGNals:SOURce:CURRent<i>](#)" on page 668
  - "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VMAXimum:ONOFF:ON

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:VMAXimum:ONOFF:ON <value>[suffix]`

`<value>` ::= Maximum expected input Voltage in NR3 format

`[suffix]` ::= {v | mV}

The :POWER:SIGNals:VMAXimum:ONOFF:ON command specifies the maximum expected input voltage. This value is used to set the vertical scale of the channel probing voltage for turn on analysis.

**Query Syntax**    `:POWER:SIGNals:VMAXimum:ONOFF:ON?`

The :POWER:SIGNals:VMAXimum:ONOFF:ON query returns the expected maximum input voltage setting.

**Return Format**    `<value><NL>`

`<value>` ::= Maximum expected input Voltage in NR3 format

- See Also**
- "[:POWER:ONOFF:APPLy](#)" on page 641
  - "[:POWER:SIGNals:AUTosetup](#)" on page 651
  - "[:POWER:SIGNals:DURation:ONOFF:ON](#)" on page 657
  - "[:POWER:SIGNals:IEXPected](#)" on page 660
  - "[:POWER:SIGNals:OVERshoot](#)" on page 661
  - "[:POWER:SIGNals:VSTeady:ONOFF:ON](#)" on page 666
  - "[:POWER:SIGNals:SOURce:CURRent<i>](#)" on page 668
  - "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VSteady:ONOFF:OFF

**N** (see [page 1334](#))

**Command Syntax** :POWER:SIGNals:VSteady:ONOFF:OFF <value>[suffix]  
 <value> ::= Expected steady state output Voltage value in NR3 format  
 [suffix] ::= {V | mV}

The :POWER:SIGNals:VSteady:ONOFF:OFF command specifies the expected steady state output DC voltage of the power supply for turn off analysis.

**Query Syntax** :POWER:SIGNals:VSteady:ONOFF:OFF?

The :POWER:SIGNals:VSteady:ONOFF:OFF query returns the expected steady state voltage setting.

**Return Format** <value><NL>  
 <value> ::= Expected steady state output Voltage value in NR3 format

**See Also**

- "[:POWER:ONOFF:APPLY](#)" on page 641
- "[:POWER:SIGNals:AUTosetup](#)" on page 651
- "[:POWER:SIGNals:DURation:ONOFF:OFF](#)" on page 656
- "[:POWER:SIGNals:IEXPected](#)" on page 660
- "[:POWER:SIGNals:OVERshoot](#)" on page 661
- "[:POWER:SIGNals:VMAXimum:ONOFF:OFF](#)" on page 663
- "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
- "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VSteady:ONOFF:ON

**N** (see [page 1334](#))

**Command Syntax**    `:POWER:SIGNals:VSteady:ONOFF:ON <value>[suffix]`

`<value>` ::= Expected steady state output Voltage value in NR3 format

`[suffix]` ::= {V | mV}

The :POWER:SIGNals:VSteady:ONOFF:ON command specifies the expected steady state output DC voltage of the power supply for turn on analysis.

**Query Syntax**    `:POWER:SIGNals:VSteady:ONOFF:ON?`

The :POWER:SIGNals:VSteady:ONOFF:ON query returns the expected steady state voltage setting.

**Return Format**    `<value><NL>`

`<value>` ::= Expected steady state output Voltage value in NR3 format

**See Also**

- "[:POWER:ONOFF:APPLy](#)" on page 641
- "[:POWER:SIGNals:AUTosetup](#)" on page 651
- "[:POWER:SIGNals:DURation:ONOFF:ON](#)" on page 657
- "[:POWER:SIGNals:IEXPected](#)" on page 660
- "[:POWER:SIGNals:OVERshoot](#)" on page 661
- "[:POWER:SIGNals:VMAXimum:ONOFF:ON](#)" on page 664
- "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
- "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:VSteady:TRANSient

**N** (see [page 1334](#))

**Command Syntax**

```
:POWER:SIGNals:VSteady:TRANSient <value>[suffix]
<value> ::= Expected steady state output Voltage value in NR3 format
[suffix] ::= {V | mV}
```

The :POWER:SIGNals:VSteady:TRANSient command specifies the expected steady state output DC voltage of the power supply for transient response analysis.

This value is used along with the overshoot percentage to specify the settling band for the transient response and to adjust the vertical scale of the oscilloscope.

**Query Syntax**

```
:POWER:SIGNals:VSteady:TRANSient?
```

The :POWER:SIGNals:VSteady:TRANSient query returns the expected steady state voltage setting.

**Return Format**

```
<value><NL>
<value> ::= Expected steady state output Voltage value in NR3 format
```

- See Also**
- "[:POWER:TRANSient:APPLy](#)" on page 678
  - "[:POWER:SIGNals:AUTosetup](#)" on page 651
  - "[:POWER:SIGNals:DURation:TRANSient](#)" on page 659
  - "[:POWER:SIGNals:IEXPected](#)" on page 660
  - "[:POWER:SIGNals:OVERshoot](#)" on page 661
  - "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668
  - "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:SOURce:CURRent<i>

**N** (see [page 1334](#))

**Command Syntax**

```
:POWER:SIGNals:SOURce:CURRent<i> <source>
    <i> ::= 1, 2 in NR1 format
    <source> ::= CHANnel<n>
    <n> ::= 1 to (# analog channels) in NR1 format
```

The :POWER:SIGNals:SOURce:CURRent<i> command specifies the first, and perhaps second, current source channel to be used in the power analysis.

**Query Syntax**

```
:POWER:SIGNals:SOURce:CURRent<i>?
```

The :POWER:SIGNals:SOURce:CURRent<i> query returns the current source channel setting.

**Return Format**

```
<source><NL>
    <source> ::= CHANnel<n>
    <n> ::= 1 to (# analog channels) in NR1 format
```

**See Also**

- "[:POWER:SIGNals:AUTosetup](#)" on page 651
- "[:POWER:SIGNals:CYCLES:HARMonics](#)" on page 652
- "[:POWER:SIGNals:CYCLES:QUALity](#)" on page 653
- "[:POWER:SIGNals:DURATION:EFFiciency](#)" on page 654
- "[:POWER:SIGNals:DURATION:MODulation](#)" on page 655
- "[:POWER:SIGNals:DURATION:ONOFF:OFF](#)" on page 656
- "[:POWER:SIGNals:DURATION:ONOFF:ON](#)" on page 657
- "[:POWER:SIGNals:DURATION:RIPPLE](#)" on page 658
- "[:POWER:SIGNals:DURATION:TRANSient](#)" on page 659
- "[:POWER:SIGNals:IEXPected](#)" on page 660
- "[:POWER:SIGNals:OVERshoot](#)" on page 661
- "[:POWER:SIGNals:VMAXimum:INRush](#)" on page 662
- "[:POWER:SIGNals:VMAXimum:ONOFF:OFF](#)" on page 663
- "[:POWER:SIGNals:VMAXimum:ONOFF:ON](#)" on page 664
- "[:POWER:SIGNals:VSTeady:ONOFF:OFF](#)" on page 665
- "[:POWER:SIGNals:VSTeady:ONOFF:ON](#)" on page 666
- "[:POWER:SIGNals:VSTeady:TRANSient](#)" on page 667
- "[:POWER:SIGNals:SOURce:VOLTage<i>](#)" on page 669

## :POWER:SIGNals:SOURce:VOLTage<i>

**N** (see [page 1334](#))

**Command Syntax**

```
:POWER:SIGNals:SOURce:VOLTage<i> <source>
    <i> ::= 1, 2 in NR1 format
    <source> ::= CHANnel<n>
    <n> ::= 1 to (# analog channels) in NR1 format
```

The :POWER:SIGNals:SOURce:VOLTage<i> command specifies the first, and perhaps second, voltage source channel to be used in the power analysis.

**Query Syntax**

```
:POWER:SIGNals:SOURce:VOLTage<i>?
```

The :POWER:SIGNals:SOURce:VOLTage<i> query returns the voltage source channel setting.

**Return Format**

```
<source><NL>
    <source> ::= CHANnel<n>
    <n> ::= 1 to (# analog channels) in NR1 format
```

**See Also**

- "[:POWER:SIGNals:AUTosetup](#)" on page 651
- "[:POWER:SIGNals:CYCLES:HARMonics](#)" on page 652
- "[:POWER:SIGNals:CYCLES:QUALity](#)" on page 653
- "[:POWER:SIGNals:DURation:EFFiciency](#)" on page 654
- "[:POWER:SIGNals:DURation:MODulation](#)" on page 655
- "[:POWER:SIGNals:DURation:ONOFF:OFF](#)" on page 656
- "[:POWER:SIGNals:DURation:ONOFF:ON](#)" on page 657
- "[:POWER:SIGNals:DURation:RIPPLE](#)" on page 658
- "[:POWER:SIGNals:DURation:TRANSient](#)" on page 659
- "[:POWER:SIGNals:IEXPected](#)" on page 660
- "[:POWER:SIGNals:OVERshoot](#)" on page 661
- "[:POWER:SIGNals:VMAXimum:INRush](#)" on page 662
- "[:POWER:SIGNals:VMAXimum:ONOFF:OFF](#)" on page 663
- "[:POWER:SIGNals:VMAXimum:ONOFF:ON](#)" on page 664
- "[:POWER:SIGNals:VSTeady:ONOFF:OFF](#)" on page 665
- "[:POWER:SIGNals:VSTeady:ONOFF:ON](#)" on page 666
- "[:POWER:SIGNals:VSTeady:TRANSient](#)" on page 667
- "[:POWER:SIGNals:SOURce:CURREnt<i>](#)" on page 668

## :POWer:SLEW:APPLy

 (see [page 1334](#))

**Command Syntax** :POWer:SLEW:APPLy

The :POWer:SLEW:APPLy command applies the slew rate analysis.

**See Also** • [":POWer:SLEW:SOURce"](#) on page 671

**:POWer:SLEW:SOURce**

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:SLEW:SOURce <source>`  
                      `<source> ::= {V | I}`

The :POWer:SLEW:SOURce command selects either the voltage source or the current source as the source for the slew rate analysis.

**Query Syntax**    `:POWer:SLEW:SOURce?`

The :POWer:SLEW:SOURce query returns the selected source for the slew rate analysis.

**Return Format**    `<source><NL>`  
                      `<source> ::= {V | I}`

**See Also**    • [":POWer:SLEW:APPLy"](#) on page 670

## :POWer:SWITch:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:SWITch:APPLy

The :POWer:SWITch:APPLy command applies the switching loss analysis using the conduction calculation method, V reference, and I reference settings.

**See Also**

- "[:POWer:SWITch:CONDuction](#)" on page 673
- "[:POWer:SWITch:IREFerence](#)" on page 674
- "[:POWer:SWITch:RDS](#)" on page 675
- "[:POWer:SWITch:VCE](#)" on page 676
- "[:POWer:SWITch:VREFerence](#)" on page 677
- "[:MEASure:ELOSSs](#)" on page 552
- "[:MEASure:PLOSSs](#)" on page 559

## :POWer:SWITch:CONDuction

**N** (see [page 1334](#))

**Command Syntax**    `:POWer:SWITch:CONDuction <conduction>`  
`<conduction> ::= {WAVeform | RDS | VCE}`

The :POWer:SWITch:CONDuction command specifies the conduction calculation method:

- WAVeform – The Power waveform uses the original voltage waveform data, and the calculation is:  $P = V \times I$
- RDS – Rds(on) – The Power waveform includes error correction:
  - In the On Zone (where the voltage level is below V Ref) – the Power calculation is:  $P = I_{d2} \times R_{ds(on)}$   
Specify Rds(on) using the :POWer:SWITch:RDS command.
  - In the Off Zone (where the current level is below I Ref) – the Power calculation is:  $P = 0$  Watt.
- VCE – Vce(sat) – The Power waveform includes error correction:
  - In the On Zone (where the voltage level is below V Ref) – the Power calculation is:  $P = V_{ce(sat)} \times I_c$   
Specify Vce(sat) using the :POWer:SWITch:VCE command.
  - In the Off Zone (where the current level is below I Ref) – the Power calculation is:  $P = 0$  Watt.

**Query Syntax**    `:POWer:SWITch:CONDuction?`

The :POWer:SWITch:CONDuction query returns the conduction calculation method.

**Return Format**    `<conduction><NL>`  
`<conduction> ::= {WAV | RDS | VCE}`

**See Also**    [":POWer:SWITch:APPLy"](#) on page 672  
[":POWer:SWITch:IREFerence"](#) on page 674  
[":POWer:SWITch:RDS"](#) on page 675  
[":POWer:SWITch:VCE"](#) on page 676  
[":POWer:SWITch:VREFerence"](#) on page 677

**:POWer:SWITch:IREFerence****N** (see [page 1334](#))

**Command Syntax**    `:POWer:SWITch:IREFerence <percent>`  
`<percent> ::= percent in NR1 format`

The :POWer:SWITch:IREFerence command to specify the current switching level for the start of switching edges. The value is in percentage of the maximum switch current.

You can adjust this value to ignore noise floors or null offset that is difficult to eliminate in current probes.

This value specifies the threshold that is used to determine the switching edges.

**Query Syntax**    `:POWer:SWITch:IREFerence?`

The :POWer:SWITch:IREFerence query returns the current switching level percent value.

**Return Format**    `<percent><NL>`  
`<percent> ::= percent in NR1 format`

**See Also**

- "[":POWer:SWITch:APPLy](#)" on page 672
- "[":POWer:SWITch:CONDuction](#)" on page 673
- "[":POWer:SWITch:RDS](#)" on page 675
- "[":POWer:SWITch:VCE](#)" on page 676
- "[":POWer:SWITch:VREFerence](#)" on page 677

## :POWer:SWITch:RDS

**N** (see [page 1334](#))

**Command Syntax** :POWer:SWITch:RDS <value>[suffix]  
 <value> ::= Rds(on) value in NR3 format  
 [suffix] ::= {OHM | mOHM}

The :POWer:SWITch:RDS command specifies the Rds(on) value when the RDS conduction calculation method is chosen (by :POWer:SWITch:CONDuction).

**Query Syntax** :POWer:SWITch:RDS?  
 The :POWer:SWITch:RDS query returns the Rds(on) value.

**Return Format** <value><NL>  
 <value> ::= Rds(on) value in NR3 format

**See Also**

- "[:POWer:SWITch:APPLy](#)" on page 672
- "[:POWer:SWITch:CONDuction](#)" on page 673
- "[:POWer:SWITch:IREFerence](#)" on page 674
- "[:POWer:SWITch:VCE](#)" on page 676
- "[:POWer:SWITch:VREFerence](#)" on page 677

**:POWer:SWITch:VCE****N** (see [page 1334](#))**Command Syntax**    `:POWer:SWITch:VCE <value>[suffix]`    `<value> ::= Vce(sat) value in NR3 format`    `[suffix] ::= {V | mV}`

The :POWer:SWITch:VCE command specifies the Vce(sat) value when the VCE conduction calculation method is chosen (by :POWer:SWITch:CONDuction).

**Query Syntax**    `:POWer:SWITch:VCE?`

The :POWer:SWITch:VCE query returns the Vce(sat) value.

**Return Format**    `<value><NL>`    `<value> ::= Vce(sat) value in NR3 format`

- See Also**
- "[:POWer:SWITch:APPLy](#)" on page 672
  - "[:POWer:SWITch:CONDuction](#)" on page 673
  - "[:POWer:SWITch:IREFerence](#)" on page 674
  - "[:POWer:SWITch:RDS](#)" on page 675
  - "[:POWer:SWITch:VREFerence](#)" on page 677

## :POWer:SWITch:VREFerence



(see [page 1334](#))

**Command Syntax**    `:POWer:SWITch:VREFerence <percent>`  
`<percent> ::= percent in NR1 format`

The :POWer:SWITch:VREFerence command to specify the voltage switching level for the switching edges. The value is in percentage of the maximum switch voltage.

You can adjust this value to ignore noise floors.

This value specifies the threshold that is used to determine the switching edges.

**Query Syntax**    `:POWer:SWITch:VREFerence?`

The :POWer:SWITch:VREFerence query returns the voltage switching level percent value.

**Return Format**    `<percent><NL>`  
`<percent> ::= percent in NR1 format`

**See Also**

- "[:POWer:SWITch:APPLy](#)" on page 672
- "[:POWer:SWITch:CONDuction](#)" on page 673
- "[:POWer:SWITch:IREFerence](#)" on page 674
- "[:POWer:SWITch:RDS](#)" on page 675
- "[:POWer:SWITch:VCE](#)" on page 676

## :POWer:TRANsient:APPLy

**N** (see [page 1334](#))

**Command Syntax** :POWer:TRANsient:APPLy

The :POWer:TRANsient:APPLy command applies the transient analysis using the initial current and new current settings.

**See Also**

- "[:POWer:TRANsient:EXIT](#)" on page 679
- "[:POWer:TRANsient:IINitial](#)" on page 680
- "[:POWer:TRANsient:INEW](#)" on page 681
- "[:POWer:TRANsient:NEXT](#)" on page 682
- "[:MEASure:TRESPonse](#)" on page 564

## :POWer:TRANsient:EXIT

**N** (see [page 1334](#))

**Command Syntax** :POWer:TRANsient:EXIT

The :POWer:TRANsient:EXIT command exits (stops) the transient analysis.

This command is equivalent to pressing the **Exit** softkey on the oscilloscope front panel during the analysis.

**See Also**

- "[":POWer:TRANsient:APPLy](#)" on page 678
- "[":POWer:TRANsient:IINitial](#)" on page 680
- "[":POWer:TRANsient:INEW](#)" on page 681
- "[":POWer:TRANsient:NEXT](#)" on page 682

**:POWER:TRANsient:IINitial****N** (see [page 1334](#))

**Command Syntax**    `:POWER:TRANsient:IINitial <value>[suffix]`  
`<value> ::= Initial current value in NR3 format`  
`[suffix] ::= {A | mA}`

The :POWER:TRANsient:IINitial command to specify the initial load current value. The initial load current will be used as a reference and to trigger the oscilloscope.

**Query Syntax**    `:POWER:TRANsient:IINitial?`

The :POWER:TRANsient:IINitial query returns the initial load current value.

**Return Format**    `<value><NL>`  
`<value> ::= Initial current value in NR3 format`

**See Also**

- "[:POWER:SIGNals:VSTeady:TRANsient](#)" on page 667
- "[:POWER:TRANsient:APPLY](#)" on page 678
- "[:POWER:TRANsient:EXIT](#)" on page 679
- "[:POWER:TRANsient:INew](#)" on page 681
- "[:POWER:TRANsient:NEXT](#)" on page 682

## :POWer:TRANsient:INew

**N** (see [page 1334](#))

**Command Syntax** :POWer:TRANsient:INew <value>[suffix]

<value> ::= New current value in NR3 format

[suffix] ::= {A | mA}

The :POWer:TRANsient:INew command to specify the new load current value. The new load current will be used as a reference and to trigger the oscilloscope.

**Query Syntax** :POWer:TRANsient:INew?

The :POWer:TRANsient:INew query returns the new load current value.

**Return Format** <value><NL>

<value> ::= New current value in NR3 format

- See Also**
- "[:POWer:TRANsient:APPLy](#)" on page 678
  - "[:POWer:TRANsient:EXIT](#)" on page 679
  - "[:POWer:TRANsient:IINitial](#)" on page 680
  - "[:POWer:TRANsient:NEXT](#)" on page 682

## :POWer:TRANsient:NEXT

**N** (see [page 1334](#))

**Command Syntax** :POWer:TRANsient:NEXT

The :POWer:TRANsient:NEXT command goes to the next step of the transient analysis.

This command is equivalent to pressing the **Next** softkey on the oscilloscope front panel when prompted during the analysis.

**See Also**

- "[":POWer:TRANsient:APPLy](#)" on page 678
- "[":POWer:TRANsient:EXIT](#)" on page 679
- "[":POWer:TRANsient:IINitial](#)" on page 680
- "[":POWer:TRANsient:INEW](#)" on page 681

## 27 :RECall Commands

Recall previously saved oscilloscope setups, reference waveforms, and masks.

**Table 108:**:RECall Commands Summary

Command	Query	Options and Query Returns
:RECall:ARbitrary[:START] [<file_spec>] [, <column>] [, <wavegen_id>] (see page 685)	n/a	<p>&lt;file_spec&gt; ::= {&lt;internal_loc&gt;   &lt;file_name&gt;}</p> <p>&lt;column&gt; ::= Column in CSV file to load. Column number starts from 1.</p> <p>&lt;internal_loc&gt; ::= 0-3; an integer in NR1 format</p> <p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>&lt;wavegen_id&gt; ::= WGEN1</p>
:RECall:DBC[:START] [<file_name>] [, <serialbus>] (see page 686)	n/a	<p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>If extension included in file name, it must be ".dbc".</p> <p>&lt;serialbus&gt; ::= {SBUS&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# of serial bus) in NR1 format</p>
:RECall:FILEname <base_name> (see page 687)	:RECall:FILEname? (see page 687)	<p>&lt;base_name&gt; ::= quoted ASCII string</p>
:RECall:LDF[:START] [<file_name>] [, <serialbus>] (see page 688)	n/a	<p>&lt;file_name&gt; ::= quoted ASCII string</p> <p>If extension included in file name, it must be ".ldf".</p> <p>&lt;serialbus&gt; ::= {SBUS&lt;n&gt;}</p> <p>&lt;n&gt; ::= 1 to (# of serial bus) in NR1 format</p>

**Table 108:** RECall Commands Summary (continued)

Command	Query	Options and Query Returns
:RECall:MASK[:START] [<file_spec>] (see <a href="#">page 689</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-3; an integer in NR1 format <file_name> ::= quoted ASCII string
:RECall:PWD <path_name> (see <a href="#">page 690</a> )	:RECall:PWD? (see <a href="#">page 690</a> )	<path_name> ::= quoted ASCII string
:RECall:SETup[:START] [<file_spec>] (see <a href="#">page 691</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:RECall:WMEMory<r>[:S TART] [<file_name>] (see <a href="#">page 692</a> )	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format <file_name> ::= quoted ASCII string If extension included in file name, it must be ".h5".

**Introduction to :RECall Commands** The :RECall subsystem provides commands to recall previously saved oscilloscope setups, reference waveforms, and masks.

### Reporting the Setup

Use :RECall? to query setup information for the RECall subsystem.

### Return Format

The following is a sample response from the :RECall? query. In this case, the query was issued following the \*RST command.

```
:REC:FIL "scope_0"
```

## :RECall:ARBitrary[:STARt]

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:RECall:ARBitrary[:STARt] [&lt;file_spec&gt; [, &lt;column&gt;] [, &lt;wavegen_id&gt;]</code>
	<code>&lt;file_spec&gt; ::= {&lt;internal_loc&gt;   &lt;file_name&gt;}</code>
	<code>&lt;column&gt; ::= Column in CSV file to load. Column number starts from 1.</code>
	<code>&lt;wavegen_id&gt; ::= WGEN1 - specifies which wavegen</code>
	<code>&lt;internal_loc&gt; ::= 0-3; an integer in NR1 format</code>
	<code>&lt;file_name&gt; ::= quoted ASCII string</code>

The :RECall:ARBitrary[:STARt] command recalls an arbitrary waveform.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".csv".

For internal locations, the <column> parameter is ignored.

For external (USB storage device) files, the column parameter is optional. If no <column> parameter is entered, and it is a 2-column file, the 2nd column (assumed to be voltage) is automatically selected. If the <column> parameter is entered, and that column does not exist in the file, the operation fails.

When recalling arbitrary waveforms (from an external USB storage device) that were not saved from the oscilloscope, be aware that the oscilloscope uses a maximum of 8192 points for an arbitrary waveform. For more efficient recalls, make sure your arbitrary waveforms are 8192 points or less.

The <wavegen\_id> parameter specifies which waveform generator to recall the arbitrary waveform into.

### See Also

- ["Introduction to :RECall Commands"](#) on page 684
- [":RECall:FILENAME"](#) on page 687
- [":RECall:PWD"](#) on page 690
- [":SAVE:ARBITRARY\[:STARt\]"](#) on page 697

## :RECall:DBC[:STARt]

**N** (see [page 1334](#))

**Command Syntax**    `:RECall:DBC[:STARt] [<file_name> [, <serialbus>]]`

`<file_name>` ::= quoted ASCII string

`<serialbus>` ::= {SBUS<n>}

`<n>` ::= 1 to (# of serial bus) in NR1 format

The :RECall:DBC[:STARt] command loads a CAN DBC (communication database) symbolic data file into the oscilloscope.

### NOTE

If a file extension is provided as part of a specified `<file_name>`, it must be ".dbc".

The `<serialbus>` parameter specifies which serial decode waveform the CAN symbolic data will be loaded for.

### See Also

- "[Introduction to :RECall Commands](#)" on page 684
- "[:RECall:FILEname](#)" on page 687
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge](#)" on page 772
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:SIGNAl](#)" on page 773
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:VALUe](#)" on page 774
- "[:SEARch:SERial:CAN:MODE](#)" on page 957
- "[:SEARch:SERial:CAN:SYMBOLic:MESSAge](#)" on page 963
- "[:SEARch:SERial:CAN:SYMBOLic:SIGNAl](#)" on page 964
- "[:SEARch:SERial:CAN:SYMBOLic:VALUe](#)" on page 965

## :RECall:FILEname

**N** (see [page 1334](#))

**Command Syntax**    `:RECall:FILEname <base_name>`  
`<base_name> ::= quoted ASCII string`

The :RECall:FILEname command specifies the source for any RECall operations.

### NOTE

This command specifies a file's base name only, without path information or an extension.

---

**Query Syntax**    `:RECall:FILEname?`

The :RECall:FILEname? query returns the current RECall filename.

**Return Format**    `<base_name><NL>`  
`<base_name> ::= quoted ASCII string`

**See Also**

- ["Introduction to :RECall Commands" on page 684](#)
- [":RECall:SETup\[:STARt\]" on page 691](#)
- [":SAVE:FILEname" on page 698](#)

## :RECall:LDF[:START]

**N** (see [page 1334](#))

**Command Syntax**    `:RECall:LDF[:START] [<file_name> [, <serialbus>]]`

`<file_name>` ::= quoted ASCII string

`<serialbus>` ::= {SBUS<n>}

`<n>` ::= 1 to (# of serial bus) in NR1 format

The :RECall:LDF[:STARt] command loads a LIN description file (LDF) symbolic data file into the oscilloscope.

### NOTE

If a file extension is provided as part of a specified `<file_name>`, it must be ".ldf".

The `<serialbus>` parameter specifies which serial decode waveform the LIN symbolic data will be loaded for.

**See Also**

- "[Introduction to :RECall Commands](#)" on page 684
- "[:RECall:FILEname](#)" on page 687
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:TRIGger:SYMBOLic:FRAME](#)" on page 838
- "[:SBUS<n>:LIN:TRIGger:SYMBOLic:SIGNAl](#)" on page 839
- "[:SBUS<n>:LIN:TRIGger:SYMBOLic:VALue](#)" on page 840
- "[:SEARch:SERial:LIN:MODE](#)" on page 987
- "[:SEARch:SERial:LIN:SYMBOLic:FRAME](#)" on page 991
- "[:SEARch:SERial:LIN:SYMBOLic:SIGNAl](#)" on page 992
- "[:SEARch:SERial:LIN:SYMBOLic:VALue](#)" on page 993

## :RECall:MASK[:STARt]

**N** (see [page 1334](#))

### Command Syntax

```
:RECall:MASK[:STARt] [<file_spec>]  
<file_spec> ::= {<internal_loc> | <file_name>}  
<internal_loc> ::= 0-3; an integer in NR1 format  
<file_name> ::= quoted ASCII string
```

The :RECall:MASK[:STARt] command recalls a mask.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".msk".

### See Also

- "[Introduction to :RECall Commands](#)" on page 684
- "[":RECall:FILEname](#)" on page 687
- "[":RECall:PWD](#)" on page 690
- "[":SAVE:MASK\[:STARt\]](#)" on page 705
- "[":MTESt:DATA](#)" on page 582

## :RECall:PWD

**N** (see [page 1334](#))

**Command Syntax**    `:RECall:PWD <path_name>`

`<path_name> ::= quoted ASCII string`

The :RECall:PWD command sets the present working directory for recall operations.

**NOTE**

Presently, the internal "/User Files" directory you see in the oscilloscope's front panel user interface is the "\Agilent Flash" directory you see in the remote interface.

**Query Syntax**    `:RECall:PWD?`

The :RECall:PWD? query returns the currently set working directory for recall operations.

**Return Format**    `<path_name><NL>`

`<path_name> ::= quoted ASCII string`

**See Also**

- "[Introduction to :RECall Commands](#)" on page 684
- "[":SAVE:PWD"](#) on page 708

## :RECall:SETUp[:STARt]

**N** (see [page 1334](#))

**Command Syntax** :RECall:SETUp [:STARt] [<file\_spec>]

<file\_spec> ::= {<internal\_loc> | <file\_name>}

<internal\_loc> ::= 0-9; an integer in NR1 format

<file\_name> ::= quoted ASCII string

The :RECall:SETUp[:STARt] command recalls an oscilloscope setup.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".scp".

---

### See Also

- "[Introduction to :RECall Commands](#)" on page 684
- "[:RECall:FILEname](#)" on page 687
- "[:RECall:PWD](#)" on page 690
- "[:SAVE\[:SETUp\[:STARt\]\]](#)" on page 715

**:RECall:WMEMory<r>[:STARt]****N** (see [page 1334](#))

**Command Syntax**    `:RECall:WMEMory<r>[:STARt] [<file_name>]`  
`<r> ::= 1 to (# ref waveforms) in NR1 format`  
`<file_name> ::= quoted ASCII string`

The :RECall:WMEMory<r>[:STARt] command recalls a reference waveform.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, it must be ".h5".

---

**See Also**

- "[Introduction to :RECall Commands](#)" on page 684
- "[:RECall:FILENAME](#)" on page 687
- "[:SAVE:WMEMORY\[:STARt\]](#)" on page 722

## 28 :SAVE Commands

Save oscilloscope setups, screen images, and data. See "[Introduction to :SAVE Commands](#)" on page 695.

**Table 109**:SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:ARbitrary[:STARt] [<file_spec>] [, <wavegen_id>] (see <a href="#">page 697</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-3; an integer in NR1 format <file_name> ::= quoted ASCII string <wavegen_id> ::= WGEN1
:SAVE:FILEname <base_name> (see <a href="#">page 698</a> )	:SAVE:FILEname? (see <a href="#">page 698</a> )	<base_name> ::= quoted ASCII string
:SAVE:IMAGE[:START] [<file_name>] (see <a href="#">page 699</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:IMAGE:FACTors {{0   OFF}   {1   ON}} (see <a href="#">page 700</a> )	:SAVE:IMAGE:FACTors? (see <a href="#">page 700</a> )	{0   1}
:SAVE:IMAGE:FORMAT <format> (see <a href="#">page 701</a> )	:SAVE:IMAGE:FORMAT? (see <a href="#">page 701</a> )	<format> ::= {{BMP   BMP24bit}   BMP8bit   PNG   NONE}
:SAVE:IMAGE:INKSaver {{0   OFF}   {1   ON}} (see <a href="#">page 702</a> )	:SAVE:IMAGE:INKSaver? (see <a href="#">page 702</a> )	{0   1}
:SAVE:IMAGE:PALETTE <palette> (see <a href="#">page 703</a> )	:SAVE:IMAGE:PALETTE? (see <a href="#">page 703</a> )	<palette> ::= {COLOR   GRAYscale}
:SAVE:LISTER[:START] [<file_name>] (see <a href="#">page 704</a> )	n/a	<file_name> ::= quoted ASCII string

**Table 109**:SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:MASK[:START] [<file_spec>] (see <a href="#">page 705</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-3; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:MULTi[:START] [<file_name>] (see <a href="#">page 706</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:POWer[:START] [<file_name>] (see <a href="#">page 707</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:PWD <path_name> (see <a href="#">page 708</a> )	:SAVE:PWD? (see <a href="#">page 708</a> )	<path_name> ::= quoted ASCII string
:SAVE:RESults[:START] [<file_spec>] (see <a href="#">page 709</a> )	n/a	<file_name> ::= quoted ASCII string
:SAVE:RESults:FORMAT: CURSor {{0   OFF}   {1   ON}} (see <a href="#">page 710</a> )	:SAVE:RESults:FORMAT: CURSor? (see <a href="#">page 710</a> )	{0   1}
:SAVE:RESults:FORMAT: MASK {{0   OFF}   {1   ON}} (see <a href="#">page 711</a> )	:SAVE:RESults:FORMAT: MASK? (see <a href="#">page 711</a> )	{0   1}
:SAVE:RESults:FORMAT: MEASurement {{0   OFF}   {1   ON}} (see <a href="#">page 712</a> )	:SAVE:RESults:FORMAT: MEASurement? (see <a href="#">page 712</a> )	{0   1}
:SAVE:RESults:FORMAT: SEARch {{0   OFF}   {1   ON}} (see <a href="#">page 713</a> )	:SAVE:RESults:FORMAT: SEARch? (see <a href="#">page 713</a> )	{0   1}
:SAVE:RESults:FORMAT: SEGmented {{0   OFF}   {1   ON}} (see <a href="#">page 714</a> )	:SAVE:RESults:FORMAT: SEGmented? (see <a href="#">page 714</a> )	{0   1}
:SAVE:SETup[:START] [<file_spec>] (see <a href="#">page 715</a> )	n/a	<file_spec> ::= {<internal_loc>   <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string

**Table 109:** SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:WAVeform[:STARt] [<file_name>] (see page 716)	n/a	<file_name> ::= quoted ASCII string
:SAVE:WAVeform:FORMAT <format> (see page 717)	:SAVE:WAVeform:FORMAT? (see page 717)	<format> ::= {ASCIixy   CSV   BINary   NONE}
:SAVE:WAVeform:LENGTH <length> (see page 718)	:SAVE:WAVeform:LENGTH? (see page 718)	<length> ::= 100 to max. length; an integer in NR1 format
:SAVE:WAVeform:LENGTH :MAX {{0   OFF}   {1   ON}} (see page 719)	:SAVE:WAVeform:LENGTH :MAX? (see page 719)	{0   1}
:SAVE:WAVeform:SEGMed <option> (see page 720)	:SAVE:WAVeform:SEGMed? (see page 720)	<option> ::= {ALL   CURRent}
:SAVE:WMEMory:SOURce <source> (see page 721)	:SAVE:WMEMory:SOURce? (see page 721)	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCTION&lt;m&gt;   MATH&lt;m&gt;   WMEMory&lt;r&gt;}</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p> <p>&lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</p> <p>NOTE: Only ADD or SUBtract math operations can be saved as reference waveforms.</p> <p>&lt;return_value&gt; ::= &lt;source&gt;</p>
:SAVE:WMEMory[:STARt] [<file_name>] (see page 722)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".h5".

**Introduction to :SAVE Commands** The :SAVE subsystem provides commands to save oscilloscope setups, screen images, and data.

:SAV is an acceptable short form for :SAVE.

### Reporting the Setup

Use :SAVE? to query setup information for the SAVE subsystem.

### Return Format

The following is a sample response from the :SAVE? query. In this case, the query was issued following the \*RST command.

```
:SAVE:FIL ""; :SAVE:IMAG:AREA GRAT;FACT 0;FORM TIFF;INKS 0;PAL  
MON; :SAVE:PWD "C:/setups/"; :SAVE:WAV:FORM NONE;LENG 1000;SEGM CURR
```

## :SAVE:ARBitrary[:STARt]

**N** (see [page 1334](#))

**Command Syntax**

```
:SAVE:ARBitrary[:STARt] [<file_spec> [, <wavegen_id>]
<file_spec> ::= {<internal_loc> | <file_name>}
<internal_loc> ::= 0-3; an integer in NR1 format
<file_name> ::= quoted ASCII string
<wavegen_id> ::= WGEN1
```

The :SAVE:ARBitrary[:STARt] command saves the current arbitrary waveform to an internal location or a file on a USB storage device.

The <wavegen\_id> parameter specifies which waveform generator to save the arbitrary waveform from.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".csv".

### See Also

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:FILEname](#)" on page 698
- "[":SAVE:PWD](#)" on page 708
- "[":RECall:ARBitrary\[:STARt\]](#)" on page 685

## :SAVE:FILEname

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:FILEname <base_name>`

`<base_name> ::= quoted ASCII string`

The :SAVE:FILEname command specifies the source for any SAVE operations.

**NOTE**

This command specifies a file's base name only, without path information or an extension.

---

**Query Syntax**    `:SAVE:FILEname?`

The :SAVE:FILEname? query returns the current SAVE filename.

**Return Format**    `<base_name><NL>`

`<base_name> ::= quoted ASCII string`

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695

- "[:SAVE:IMAGe\[:STARt\]](#)" on page 699

- "[:SAVE\[:SETUp\[:STARt\]\]](#)" on page 715

- "[:SAVE:WAVEform\[:STARt\]](#)" on page 716

- "[:SAVE:PWD](#)" on page 708

- "[:RECall:FILEname](#)" on page 687

**:SAVE:IMAGe[:STARt]****N** (see [page 1334](#))

**Command Syntax**    `:SAVE:IMAGe [:STARt] [<file_name>]`  
`<file_name> ::= quoted ASCII string`

The :SAVE:IMAGe[:STARt] command saves an image.

**NOTE**

Be sure to set the :SAVE:IMAGe:FORMat before saving an image. If the format is NONE, the save image command will not succeed.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, and it does not match the extension expected by the format specified in :SAVE:IMAGe:FORMat, the format will be changed if the extension is a valid image file extension.

**NOTE**

If the extension ".bmp" is used and the current :SAVE:IMAGe:FORMat is not BMP or BMP8, the format will be changed to BMP.

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:IMAGe:FACTors](#)" on page 700
- "[":SAVE:IMAGe:FORMat](#)" on page 701
- "[":SAVE:IMAGe:INKSaver](#)" on page 702
- "[":SAVE:IMAGe:PALETTE](#)" on page 703
- "[":SAVE:FILEname](#)" on page 698

## :SAVE:IMAGe:FACTors

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:IMAGe:FACTors <factors>`  
`<factors> ::= {{OFF | 0} | {ON | 1}}`

The :SAVE:IMAGe:FACTors command controls whether the oscilloscope factors are output along with the image.

### NOTE

Factors are written to a separate file with the same path and base name but with the ".txt" extension.

**Query Syntax**    `:SAVE:IMAGe:FACTors?`

The :SAVE:IMAGe:FACTors? query returns a flag indicating whether oscilloscope factors are output along with the image.

**Return Format**    `<factors><NL>`  
`<factors> ::= {0 | 1}`

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:IMAGe\[:STARt\]](#)" on page 699
- "[":SAVE:IMAGe:FORMAT](#)" on page 701
- "[":SAVE:IMAGe:INKSaver](#)" on page 702
- "[":SAVE:IMAGe:PALETTE](#)" on page 703

## :SAVE:IMAGe:FORMAT

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:IMAGe:FORMAT <format>`

`<format> ::= {{BMP | BMP24bit} | BMP8bit | PNG}`

The :SAVE:IMAGe:FORMAT command sets the image format type.

**Query Syntax**    `:SAVE:IMAGe:FORMAT?`

The :SAVE:IMAGe:FORMAT? query returns the selected image format type.

**Return Format**    `<format><NL>`

`<format> ::= {BMP | BMP8 | PNG | NONE}`

When NONE is returned, it indicates that a waveform data file format is currently selected.

### See Also

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:IMAGe\[:STARt\]](#)" on page 699
- "[":SAVE:IMAGe:FACTors](#)" on page 700
- "[":SAVE:IMAGe:INKSaver](#)" on page 702
- "[":SAVE:IMAGe:PALETTE](#)" on page 703
- "[":SAVE:WAVEform:FORMAT](#)" on page 717

**:SAVE:IMAGe:INKSaver****N** (see [page 1334](#))

**Command Syntax**    `:SAVE:IMAGe:INKSaver <value>`  
`<value> ::= {{OFF | 0} | {ON | 1}}`

The :SAVE:IMAGe:INKSaver command controls whether the graticule colors are inverted or not.

**Query Syntax**    `:SAVE:IMAGe:INKSaver?`

The :SAVE:IMAGe:INKSaver? query returns a flag indicating whether graticule colors are inverted or not.

**Return Format**    `<value><NL>`  
`<value> ::= {0 | 1}`

**See Also**

- ["Introduction to :SAVE Commands"](#) on page 695
- [":SAVE:IMAGe\[:STARt\]"](#) on page 699
- [":SAVE:IMAGe:FACTors"](#) on page 700
- [":SAVE:IMAGe:FORMat"](#) on page 701
- [":SAVE:IMAGe:PAlette"](#) on page 703

## :SAVE:IMAGe:PALETTE

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:IMAGe:PALETTE <palette>`

`<palette> ::= {COLOR | GRAYscale}`

The :SAVE:IMAGe:PALETTE command sets the image palette color.

**Query Syntax**    `:SAVE:IMAGe:PALETTE?`

The :SAVE:IMAGe:PALETTE? query returns the selected image palette color.

**Return Format**    `<palette><NL>`

`<palette> ::= {COL | GRAY}`

**See Also**

- "Introduction to :SAVE Commands" on page 695

- "":SAVE:IMAGe[:STARt]" on page 699

- "":SAVE:IMAGe:FACTors" on page 700

- "":SAVE:IMAGe:FORMAT" on page 701

- "":SAVE:IMAGe:INKSaver" on page 702

## :SAVE:LISTER[:STARt]

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:LISTER[:STARt] [<file_name>]`  
`<file_name> ::= quoted ASCII string`

The :SAVE:LISTER[:STARt] command saves the Lister display data to a file.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".csv".

---

**See Also**

- ["Introduction to :SAVE Commands"](#) on page 695

- [":SAVE:FILEname"](#) on page 698
- [Chapter 20](#), “:LISTER Commands,” starting on page 435

**:SAVE:MASK[:STARt]****N** (see [page 1334](#))**Command Syntax**    `:SAVE:MASK[:STARt] [<file_spec>]``<file_spec> ::= {<internal_loc> | <file_name>}``<internal_loc> ::= 0-3; an integer in NR1 format``<file_name> ::= quoted ASCII string`

The :SAVE:MASK[:STARt] command saves a mask.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, it must be ".msk".

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[:SAVE:FILEname](#)" on page 698
- "[:SAVE:PWD](#)" on page 708
- "[:RECall:MASK\[:STARt\]](#)" on page 689
- "[:MTEST:DATA](#)" on page 582

**:SAVE:MULTi[:STARt]**(see [page 1334](#))

**Command Syntax**    `:SAVE:MULTi [:STARt] [<file_name>]`  
`<file_name> ::= quoted ASCII string`

The :SAVE:MULTi[:STARt] command saves multi-channel waveform data to a file. This file can be opened by the N8900A Infinium Offline oscilloscope analysis software.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, it must be ".h5".

- 
- See Also**
- "[Introduction to :SAVE Commands](#)" on page 695
  - "[":SAVE:FILENAME](#)" on page 698
  - "[":SAVE:PWD](#)" on page 708

## :SAVE:POWeR[:STARt]

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:POWeR[:STARt] [<file_name>]`  
`<file_name> ::= quoted ASCII string`

The :SAVE:POWeR[:STARt] command saves the power measurement application's current harmonics analysis results to a file.

### NOTE

If a file extension is provided as part of a specified <file\_name>, it must be ".csv".

---

### See Also

- "[Introduction to :SAVE Commands](#)" on page 695
- "[:SAVE:FILEname](#)" on page 698
- [Chapter 26, “:POWeR Commands,”](#) starting on page 607

**:SAVE:PWD****N** (see [page 1334](#))**Command Syntax**    `:SAVE:PWD <path_name>`    `<path_name> ::= quoted ASCII string`

The :SAVE:PWD command sets the present working directory for save operations.

**NOTE**

Presently, the internal "/User Files" directory you see in the oscilloscope's front panel user interface is the "\Agilent Flash" directory you see in the remote interface.

**Query Syntax**    `:SAVE:PWD?`

The :SAVE:PWD? query returns the currently set working directory for save operations.

**Return Format**    `<path_name><NL>`    `<path_name> ::= quoted ASCII string`**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:FILENAME](#)" on page 698
- "[":RECALL:PWD](#)" on page 690

## :SAVE:RESUltS:[STARt]

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:RESUltS: [STARt] [<file_spec>]  
<file_name> ::= quoted ASCII string`

The :SAVE:RESUltS:[STARt] command saves analysis results to a comma-separated values (\*.csv) file on a USB storage device.

Use the :SAVE:RESUltS:FORMat commands to specify the analysis types whose results are saved to the file.

When multiple types of analysis results are selected, they are all saved to the same file and separated by a blank line.

**See Also**

- "[":SAVE:RESUltS:FORMat:CURSor](#)" on page 710
- "[":SAVE:RESUltS:FORMat:MASK](#)" on page 711
- "[":SAVE:RESUltS:FORMat:MEASurement](#)" on page 712
- "[":SAVE:RESUltS:FORMat:SEARch](#)" on page 713
- "[":SAVE:RESUltS:FORMat:SEGmented](#)" on page 714

## :SAVE:RESults:FORMat:CURSor

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:RESults:FORMat:CURSor {{0 | OFF} | {1 | ON}}`

The :SAVE:RESults:FORMat:CURSor command specifies whether cursor values will be included when analysis results are saved.

Analysis results are saved using the :SAVE:RESults:[STARt] command.

Other :SAVE:RESults:FORMat commands specify whether other types of analysis results are also saved.

When multiple types of analysis results are saved, they are all saved to the same file and separated by a blank line.

**Query Syntax**    `:SAVE:RESults:FORMat:CURSor?`

The :SAVE:RESults:FORMat:CURSor? query returns whether cursor values will be included when analysis results are saved.

**Return Format**    `<off_on><NL>`

`{0 | 1}`

- See Also**
- "[":SAVE:RESults:\[STARt\]](#)" on page 709
  - "[":SAVE:RESults:FORMat:MASK](#)" on page 711
  - "[":SAVE:RESults:FORMat:MEASurement](#)" on page 712
  - "[":SAVE:RESults:FORMat:SEARch](#)" on page 713
  - "[":SAVE:RESults:FORMat:SEGmented](#)" on page 714

## :SAVE:RESults:FORMat:MASK

**N** (see [page 1334](#))

**Command Syntax** :SAVE:RESults:FORMat:MASK {{0 | OFF} | {1 | ON}}

The :SAVE:RESults:FORMat:MASK command specifies whether mask statistics will be included when analysis results are saved.

Analysis results are saved using the :SAVE:RESults:[STARt] command.

Other :SAVE:RESults:FORMat commands specify whether other types of analysis results are also saved.

When multiple types of analysis results are saved, they are all saved to the same file and separated by a blank line.

**Query Syntax** :SAVE:RESults:FORMat:MASK?

The :SAVE:RESults:FORMat:MASK? query returns whether mask statistics will be included when analysis results are saved.

**Return Format** <off\_on><NL>

{0 | 1}

- See Also**
- "[":SAVE:RESults:\[STARt\]](#)" on page 709
  - "[":SAVE:RESults:FORMat:CURSor](#)" on page 710
  - "[":SAVE:RESults:FORMat:MEASurement](#)" on page 712
  - "[":SAVE:RESults:FORMat:SEARch](#)" on page 713
  - "[":SAVE:RESults:FORMat:SEGmented](#)" on page 714

## :SAVE:RESults:FORMat:MEASurement

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:RESults:FORMat:MEASurement {{0 | OFF} | {1 | ON}}`

The :SAVE:RESults:FORMat:MEASurement command specifies whether measurement results will be included when analysis results are saved.

Analysis results are saved using the :SAVE:RESults:[STARt] command.

Other :SAVE:RESults:FORMat commands specify whether other types of analysis results are also saved.

When multiple types of analysis results are saved, they are all saved to the same file and separated by a blank line.

**Query Syntax**    `:SAVE:RESults:FORMat:MEASurement?`

The :SAVE:RESults:FORMat:MEASurement? query returns whether measurement results will be included when analysis results are saved.

**Return Format**    `<off_on><NL>`

`{0 | 1}`

- See Also**
- "[":SAVE:RESults:\[STARt\]](#)" on page 709
  - "[":SAVE:RESults:FORMat:CURSor](#)" on page 710
  - "[":SAVE:RESults:FORMat:MASK](#)" on page 711
  - "[":SAVE:RESults:FORMat:SEARch](#)" on page 713
  - "[":SAVE:RESults:FORMat:SEGmented](#)" on page 714

## :SAVE:RESults:FORMat:SEARch

**N** (see [page 1334](#))

**Command Syntax** :SAVE:RESults:FORMat:SEARch {{0 | OFF} | {1 | ON}}

The :SAVE:RESults:FORMat:SEARch command specifies whether found search event times will be included when analysis results are saved.

Analysis results are saved using the :SAVE:RESults:[STARt] command.

Other :SAVE:RESults:FORMat commands specify whether other types of analysis results are also saved.

When multiple types of analysis results are saved, they are all saved to the same file and separated by a blank line.

**Query Syntax** :SAVE:RESults:FORMat:SEARch?

The :SAVE:RESults:FORMat:SEARch? query returns whether found search event times will be included when analysis results are saved.

**Return Format** <off\_on><NL>

{0 | 1}

- See Also**
- "[":SAVE:RESults:\[STARt\]](#)" on page 709
  - "[":SAVE:RESults:FORMat:CURSor](#)" on page 710
  - "[":SAVE:RESults:FORMat:MASK](#)" on page 711
  - "[":SAVE:RESults:FORMat:MEASurement](#)" on page 712
  - "[":SAVE:RESults:FORMat:SEGmented](#)" on page 714

**:SAVE:RESults:FORMat:SEGmented****N** (see [page 1334](#))**Command Syntax**    `:SAVE:RESults:FORMat:SEGmented {{0 | OFF} | {1 | ON}}`

The :SAVE:RESults:FORMat:SEGmented command specifies whether segmented memory acquisition times will be included when analysis results are saved.

Analysis results are saved using the :SAVE:RESults:[STARt] command.

Other :SAVE:RESults:FORMat commands specify whether other types of analysis results are also saved.

When multiple types of analysis results are saved, they are all saved to the same file and separated by a blank line.

**Query Syntax**    `:SAVE:RESults:FORMat:SEGmented?`

The :SAVE:RESults:FORMat:SEGmented? query returns whether segmented memory acquisition times will be included when analysis results are saved.

**Return Format**    `<off_on><NL>``{0 | 1}`

- See Also**
- "[":SAVE:RESults:\[STARt\]](#)" on page 709
  - "[":SAVE:RESults:FORMat:CURSor](#)" on page 710
  - "[":SAVE:RESults:FORMat:MASK](#)" on page 711
  - "[":SAVE:RESults:FORMat:MEASurement](#)" on page 712
  - "[":SAVE:RESults:FORMat:SEARch](#)" on page 713

**:SAVE[:SETup[:STARt]]****N** (see [page 1334](#))**Command Syntax**    `:SAVE [:SETup [:STARt]] [<file_spec>]``<file_spec> ::= {<internal_loc> | <file_name>}``<internal_loc> ::= 0-9; an integer in NR1 format``<file_name> ::= quoted ASCII string`

The :SAVE[:SETup[:STARt]] command saves an oscilloscope setup.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, it must be ".scp".

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[:SAVE:FILEname](#)" on page 698
- "[:SAVE:PWD](#)" on page 708
- "[:RECall:SETup\[:STARt\]](#)" on page 691

**:SAVE:WAVEform[:STARt]****N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WAVEform[:STARt] [<file_name>]`  
                  `<file_name> ::= quoted ASCII string`

The :SAVE:WAVEform[:STARt] command saves oscilloscope waveform data to a file.

**NOTE**

Be sure to set the :SAVE:WAVEform:FORMAT before saving waveform data. If the format is NONE, the save waveform command will not succeed.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, and it does not match the extension expected by the format specified in :SAVE:WAVEform:FORMAT, the format will be changed if the extension is a valid waveform file extension.

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:WAVEform:FORMAT](#)" on page 717
- "[":SAVE:WAVEform:LENGTH](#)" on page 718
- "[":SAVE:FILENAME](#)" on page 698
- "[":RECALL:SETUP\[:STARt\]](#)" on page 691

## :SAVE:WAVEform:FORMAT

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WAVEform:FORMAT <format>`

```
<format> ::= {ASCIIxy | CSV | BINARY}
```

The :SAVE:WAVEform:FORMAT command sets the waveform data format type:

- ASCIIxy – creates comma-separated value files for each analog channel that is displayed (turned on). The proper file extension for this format is ".csv".
- CSV – creates one comma-separated value file that contains information for all analog channels that are displayed (turned on). The proper file extension for this format is ".csv".
- BINARY – creates an oscilloscope binary data format file. See the *User's Guide* for a description of this format. The proper file extension for this format is ".bin".

**Query Syntax**    `:SAVE:WAVEform:FORMAT?`

The :SAVE:WAVEform:FORMAT? query returns the selected waveform data format type.

**Return Format**    `<format><NL>`

```
<format> ::= {ASC | CSV | BIN | NONE}
```

When NONE is returned, it indicates that an image file format is currently selected.

**See Also**    ["Introduction to :SAVE Commands"](#) on page 695

- [":SAVE:WAVEform\[:START\]"](#) on page 716
- [":SAVE:WAVEform:LENGTH"](#) on page 718
- [":SAVE:IMAGE:FORMAT"](#) on page 701

**:SAVE:WAVEform:LENGth**

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WAVEform:LENGth <length>`

`<length> ::= 100 to max. length; an integer in NR1 format`

When the :SAVE:WAVEform:LENGth:MAX setting is OFF, the :SAVE:WAVEform:LENGth command sets the waveform data length (that is, the number of points saved).

When the :SAVE:WAVEform:LENGth:MAX setting is ON, the :SAVE:WAVEform:LENGth setting has no effect.

**Query Syntax**    `:SAVE:WAVEform:LENGth?`

The :SAVE:WAVEform:LENGth? query returns the current waveform data length setting.

**Return Format**    `<length><NL>`

`<length> ::= 100 to max. length; an integer in NR1 format`

**See Also**

- "Introduction to :SAVE Commands" on page 695

- "":SAVE:WAVEform:LENGth:MAX" on page 719

- "":SAVE:WAVEform[:STARt]" on page 716

- "":WAVEform:POINTs" on page 1158

- "":SAVE:WAVEform:FORMAT" on page 717

**:SAVE:WAVeform:LENGth:MAX**

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WAVeform:LENGth:MAX <setting>`  
`<setting> ::= {{OFF | 0} | {ON | 1}}`

The :SAVE:WAVeform:LENGth:MAX command specifies whether maximum number of waveform data points is saved.

When OFF, the :SAVE:WAVeform:LENGth command specifies the number of waveform data points saved.

**Query Syntax**    `:SAVE:WAVeform:LENGth:MAX?`

The :SAVE:WAVeform:LENGth:MAX? query returns the current setting.

**Return Format**    `<setting><NL>`  
`<setting> ::= {0 | 1}`

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:WAVeform\[:START\]](#)" on page 716
- "[":SAVE:WAVeform:LENGth](#)" on page 718

**:SAVE:WAVeform:SEGmented**

**N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WAVeform:SEGmented <option>`  
`<option> ::= {ALL | CURR}`

When segmented memory is used for acquisitions, the :SAVE:WAVeform:SEGmented command specifies which segments are included when the waveform is saved:

- ALL – all acquired segments are saved.
- CURR – only the currently selected segment is saved.

**Query Syntax**    `:SAVE:WAVeform:SEGmented?`

The :SAVE:WAVeform:SEGmented? query returns the current segmented waveform save option setting.

**Return Format**    `<option><NL>`  
`<option> ::= {ALL | CURR}`

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:WAVeform\[:START\]](#)" on page 716
- "[":SAVE:WAVeform:FORMAT](#)" on page 717
- "[":SAVE:WAVeform:LENGTH](#)" on page 718

## :SAVE:WMEMORY:SOURce

**N** (see [page 1334](#))

### Command Syntax

```
:SAVE:WMEMORY:SOURce <source>
<source> ::= {CHANnel<n> | FUNCtion<m> | MATH<m> | WMEMory<r>}
<n> ::= 1 to (# analog channels) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :SAVE:WMEMORY:SOURce command selects the source to be saved as a reference waveform file.

### NOTE

Only ADD or SUBtract math operations can be saved as reference waveforms.

### NOTE

MATH is an alias for FUNCtion. The query will return FUNC if the source is FUNCtion or MATH.

### Query Syntax

```
:SAVE:WMEMORY:SOURce?
```

The :SAVE:WMEMORY:SOURce? query returns the source to be saved as a reference waveform file.

### Return Format

```
<source><NL>
<source> ::= {CHAN<n> | FUNC | WMEM<r> | NONE}
```

### See Also

- "[Introduction to :SAVE Commands](#)" on page 695
- "[":SAVE:WMEMORY\[:START\]"](#) on page 722
- "[":RECall:WMEMORY<r>\[:START\]"](#) on page 692

**:SAVE:WMEMORY[:STARt]****N** (see [page 1334](#))

**Command Syntax**    `:SAVE:WMEMORY [:STARt] [<file_name>]`  
`<file_name> ::= quoted ASCII string`

The :SAVE:WMEMORY[:STARt] command saves oscilloscope waveform data to a reference waveform file.

**NOTE**

If a file extension is provided as part of a specified <file\_name>, it must be ".h5".

**See Also**

- "[Introduction to :SAVE Commands](#)" on page 695
- "[:SAVE:WMEMORY:SOURce](#)" on page 721
- "[:RECall:WMEMORY<r>\[:STARt\]](#)" on page 692

## 29 :SBUS< n > Commands

Control the modes and parameters for each serial bus decode/trigger type. See:

- "[Introduction to :SBUS< n > Commands](#)" on page 723
- "[General :SBUS< n > Commands](#)" on page 725
- "[:SBUS< n >:A429 Commands](#)" on page 728
- "[:SBUS< n >:CAN Commands](#)" on page 745
- "[:SBUS< n >:FLEXray Commands](#)" on page 775
- "[:SBUS< n >:I2S Commands](#)" on page 794
- "[:SBUS< n >:IIC Commands](#)" on page 813
- "[:SBUS< n >:LIN Commands](#)" on page 823
- "[:SBUS< n >:M1553 Commands](#)" on page 841
- "[:SBUS< n >:SENT Commands](#)" on page 848
- "[:SBUS< n >:SPI Commands](#)" on page 881
- "[:SBUS< n >:UART Commands](#)" on page 897

Introduction to  
:SBUS< n >  
Commands

The :SBUS subsystem commands control the serial decode bus viewing, mode, and other options.

**NOTE**

These commands are only valid on oscilloscope models when a serial decode option has been licensed.

The following serial bus decode/trigger types are available (see "[:TRIGger:MODE](#)" on page 1056).

- **CAN (Controller Area Network) triggering**— will trigger on CAN version 2.0A and 2.0B signals. Setup consists of connecting the oscilloscope to a CAN signal. Baud rate, signal source, and signal polarity, and type of data to trigger on can be specified. You can trigger on CAN data and identifier patterns and you can set the bit sample point.
- **I2S (Inter-IC Sound or Integrated Interchip Sound bus) triggering**— consists of connecting the oscilloscope to the serial clock, word select, and serial data lines, then triggering on a data value.

- **IIC (Inter-IC bus) triggering**— consists of connecting the oscilloscope to the serial data (SDA) line and the serial clock (SCL) line, then triggering on a stop/start condition, a restart, a missing acknowledge, or on a read/write frame with a specific device address and data value.
- **LIN (Local Interconnect Network) triggering**— will trigger on LIN sync break at the beginning of a message frame. You can trigger on Sync Break, Frame IDs, or Frame IDs and Data.
- **SPI (Serial Peripheral Interface) triggering**— consists of connecting the oscilloscope to a clock, data (MOSI or MISO), and framing signal. You can then trigger on a data pattern during a specific framing period. The serial data string can be specified to be from 4 to 64 bits long.
- **UART/RS-232 triggering** (with COMP license) – lets you trigger on RS-232 serial data.
- **SENT triggering** (with SENSOR license) – lets you trigger on SENT serial data.

**NOTE**

Two I<sub>2</sub>S buses or two SPI buses cannot be decoded on both SBUS1 and SBUS2 at the same time.

**Reporting the Setup**

Use :SBUS<n>? to query setup information for the :SBUS<n> subsystem.

**Return Format**

The following is a sample response from the :SBUS1? query. In this case, the query was issued following a \*RST command.

```
:SBUS1:DISP 0;MODE IIC;:SBUS1:IIC:ASIZ BIT7;:SBUS1:IIC:TRIG:TYPE
STAR;QUAL EQU;:SBUS1:IIC:SOUR:CLOC CHAN1;DATA
CHAN2;:SBUS1:IIC:TRIG:PATT:ADDR -1;DATA -1;DATA2 -1
```

## General :SBUS<n> Commands

**Table 110** General :SBUS<n> Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:DISPlay {{0   OFF}   {1   ON}} (see <a href="#">page 726</a> )	:SBUS<n>:DISPlay? (see <a href="#">page 726</a> )	{0   1}
:SBUS<n>:MODE <mode> (see <a href="#">page 727</a> )	:SBUS<n>:MODE? (see <a href="#">page 727</a> )	<mode> ::= {A429   CAN   FLEXray   I2S   IIC   LIN   M1553   SENT   SPI   UART}

## :SBUS<n>:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:DISPlay <display>`

`<display> ::= {{1 | ON} | {0 | OFF}}`

The :SBUS<n>:DISPlay command turns displaying of the serial decode bus on or off.

### NOTE

This command is only valid when a serial decode option has been licensed.

### NOTE

Two I2S buses or two SPI buses cannot be decoded on both SBUS1 and SBUS2 at the same time.

### Query Syntax

`:SBUS<n>:DISPlay?`

The :SBUS<n>:DISPlay? query returns the current display setting of the serial decode bus.

**Return Format**    `<display><NL>`

`<display> ::= {0 | 1}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)

    • [":CHANnel<n>:DISPlay" on page 284](#)

    • [":DIGItal<d>:DISPlay" on page 323](#)

    • [":POD<n>:DISPlay" on page 603](#)

    • [":VIEW" on page 242](#)

    • [":BLANK" on page 214](#)

    • [":STATus" on page 239](#)

**:SBUS<n>:MODE**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:MODE <mode>`

```
<mode> ::= {A429 | FLEXray | CAN | I2S | IIC | LIN | M1553 | SENT
             | SPI | UART}
```

The :SBUS<n>:MODE command determines the decode mode for the serial bus.

**NOTE**

This command is only valid when a serial decode option has been licensed.

**Query Syntax**    `:SBUS<n>:MODE?`

The :SBUS<n>:MODE? query returns the current serial bus decode mode setting.

**Return Format**    `<mode><NL>`

```
<mode> ::= {A429 | FLEX | CAN | I2S | IIC | LIN | M1553 | SENT | SPI
             | UART | NONE}
```

**Errors**    · "–241, Hardware missing" on page 1293

**See Also**    · "[Introduction to :SBUS<n> Commands](#)" on page 723  
               · "[:SBUS<n>:A429 Commands](#)" on page 728  
               · "[:SBUS<n>:CAN Commands](#)" on page 745  
               · "[:SBUS<n>:FLEXray Commands](#)" on page 775  
               · "[:SBUS<n>:I2S Commands](#)" on page 794  
               · "[:SBUS<n>:IIC Commands](#)" on page 813  
               · "[:SBUS<n>:LIN Commands](#)" on page 823  
               · "[:SBUS<n>:M1553 Commands](#)" on page 841  
               · "[:SBUS<n>:SENT Commands](#)" on page 848  
               · "[:SBUS<n>:SPI Commands](#)" on page 881  
               · "[:SBUS<n>:UART Commands](#)" on page 897

## :SBUS< n >:A429 Commands

**NOTE**

These commands are valid when the DSOX4AERO MIL-STD-1553 and ARINC 429 triggering and serial decode option (Option AERO) has been licensed.

**Table 111** :SBUS< n >:A429 Commands Summary

Command	Query	Options and Query Returns
:SBUS< n >:A429:AUTOset up (see <a href="#">page 730</a> )	n/a	n/a
:SBUS< n >:A429:BASE <base> (see <a href="#">page 731</a> )	:SBUS< n >:A429:BASE? (see <a href="#">page 731</a> )	<base> ::= {BINary   HEX}
n/a	:SBUS< n >:A429:COUNT:E RRor? (see <a href="#">page 732</a> )	<error_count> ::= integer in NR1 format
:SBUS< n >:A429:COUNT:R ESet (see <a href="#">page 733</a> )	n/a	n/a
n/a	:SBUS< n >:A429:COUNT:W ORD? (see <a href="#">page 734</a> )	<word_count> ::= integer in NR1 format
:SBUS< n >:A429:FORMAT <format> (see <a href="#">page 735</a> )	:SBUS< n >:A429:FORMAT? (see <a href="#">page 735</a> )	<format> ::= {LDSDi   LDSSm   LData}
:SBUS< n >:A429:SIGNAl <signal> (see <a href="#">page 736</a> )	:SBUS< n >:A429:SIGNAl? (see <a href="#">page 736</a> )	<signal> ::= {A   B   DIFFerential}
:SBUS< n >:A429:SOURce <source> (see <a href="#">page 737</a> )	:SBUS< n >:A429:SOURce? (see <a href="#">page 737</a> )	<source> ::= {CHANnel< n >} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS< n >:A429:SPEEd <speed> (see <a href="#">page 738</a> )	:SBUS< n >:A429:SPEEd? (see <a href="#">page 738</a> )	<speed> ::= {LOW   HIGH}
:SBUS< n >:A429:TRIGger :LABel <value> (see <a href="#">page 739</a> )	:SBUS< n >:A429:TRIGger :LABel? (see <a href="#">page 739</a> )	<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 or "0xXX" (don't care) <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}

**Table 111** :SBUS<n>:A429 Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:A429:TRIGger :PATTern:DATA <string> (see page 740)	:SBUS<n>:A429:TRIGger :PATTern:DATA? (see page 740)	<string> ::= "nn...n" where n ::= {0   1   X}, length depends on FORMat
:SBUS<n>:A429:TRIGger :PATTern:SDI <string> (see page 741)	:SBUS<n>:A429:TRIGger :PATTern:SDI? (see page 741)	<string> ::= "nn" where n ::= {0   1   X}, length always 2 bits
:SBUS<n>:A429:TRIGger :PATTern:SSM <string> (see page 742)	:SBUS<n>:A429:TRIGger :PATTern:SSM? (see page 742)	<string> ::= "nn" where n ::= {0   1   X}, length always 2 bits
:SBUS<n>:A429:TRIGger :RANGE <min>,<max> (see page 743)	:SBUS<n>:A429:TRIGger :RANGE? (see page 743)	<min> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <max> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}
:SBUS<n>:A429:TRIGger :TYPE <condition> (see page 744)	:SBUS<n>:A429:TRIGger :TYPE? (see page 744)	<condition> ::= {WSTArt   WSTOP   LABel   LBITS   PERRor   WERRor   GERRor   WGERRors   ALLerrors   LRANGE   ABITS   AOBits   AZBits}

## :SBUS< n >:A429:AUTosetup



(see [page 1334](#))

**Command Syntax**    `:SBUS<n>:A429:AUTosetup`

The `:SBUS< n >:A429:AUTosetup` command automatically sets these options for decoding and triggering on ARINC 429 signals:

- High Trigger Threshold: 3.0 V.
- Low Trigger Threshold: -3.0 V.
- Noise Reject: Off.
- Probe Attenuation: 10.0.
- Vertical Scale: 4 V/div.
- Serial Decode: On.
- Base (`:SBUS< n >:A429:BASE`): HEX.
- Word Format (`:SBUS< n >:A429:FORMat`): LDSDI (Label/SDI/Data/SSM).
- Trigger: the specified serial bus (n of `:SBUS< n >`).
- Trigger Mode (`:SBUS< n >:A429:TRIGger:TYPE`): WSTArt.

**Errors**

- "[-241, Hardware missing](#)" on page 1293

**See Also**

- "[":SBUS< n >:A429:BASE](#)" on page 731
- "[":SBUS< n >:A429:FORMat](#)" on page 735
- "[":SBUS< n >:A429:TRIGger:TYPE](#)" on page 744
- "[":Introduction to :SBUS< n > Commands](#)" on page 723
- "[":SBUS< n >:MODE](#)" on page 727
- "[":SBUS< n >:A429 Commands](#)" on page 728

## :SBUS<n>:A429:BASE

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:A429:BASE <base>`  
`<base> ::= {BINary | HEX}`

The :SBUS<n>:A429:BASE command selects between hexadecimal and binary display of the decoded data.

The BASE command has no effect on the SDI and SSM fields, which are always displayed in binary, nor the Label field, which is always displayed in octal.

**Query Syntax**    `:SBUS<n>:A429:BASE?`

The :SBUS<n>:A429:BASE? query returns the current ARINC 429 base setting.

**Return Format**    `<base><NL>`  
`<base> ::= {BIN | HEX}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:A429:FORMAT" on page 735](#)

**:SBUS<n>:A429:COUNt:ERRor****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:A429:COUNt:ERRor?`

Returns the error count.

**Return Format**    `<error_count><NL>``<error_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:A429:COUNt:RESet" on page 733](#)[":SBUS<n>:A429:COUNt:WORD" on page 734](#)["Introduction to :SBUS<n> Commands" on page 723](#)[":SBUS<n>:MODE" on page 727](#)[":SBUS<n>:A429 Commands" on page 728](#)

**:SBUS< n >:A429:COUNT:RESet**

**N** (see [page 1334](#))

**Command Syntax** :SBUS< n >:A429:COUNT:RESet

Resets the word and error counters.

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS< n >:A429:COUNT:WORD" on page 734](#)

• [":SBUS< n >:A429:COUNt:ERRor" on page 732](#)

• ["Introduction to :SBUS< n > Commands" on page 723](#)

• [":SBUS< n >:MODE" on page 727](#)

• [":SBUS< n >:A429 Commands" on page 728](#)

**:SBUS<n>:A429:COUNt:WORD****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:A429:COUNt:WORD?`

Returns the word count.

**Return Format**    `<word_count><NL>``<word_count> ::= integer in NR1 format`**Errors**    • "–241, Hardware missing" on page 1293**See Also**    • "[:SBUS<n>:A429:COUNt:RESet](#)" on page 733[• ":SBUS<n>:A429:COUNt:ERRor"](#) on page 732[• "Introduction to :SBUS<n> Commands"](#) on page 723[• ":SBUS<n>:MODE"](#) on page 727[• ":SBUS<n>:A429 Commands"](#) on page 728

## :SBUS<n>:A429:FORMAT

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:A429:FORMAT <format>`  
`<format> ::= {LDSDi | LDSSm | LDATA}`

The :SBUS<n>:A429:FORMAT command specifies the word decode format:

- LDSDi:
  - Label - 8 bits.
  - SDI - 2 bits.
  - Data - 19 bits.
  - SSM - 2 bits.
- LDSSm:
  - Label - 8 bits.
  - Data - 21 bits.
  - SSM - 2 bits.
- LDATA:
  - Label - 8 bits.
  - Data - 23 bits.

**Query Syntax**    `:SBUS<n>:A429:FORMAT?`

The :SBUS<n>:A429:FORMAT? query returns the current ARINC 429 word decode format setting.

**Return Format**    `<format><NL>`  
`<format> ::= {LDSD | LDSS | LDAT}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
 • [":SBUS<n>:MODE" on page 727](#)  
[":SBUS<n>:A429:TRIGger:PATTERn:DATA" on page 740](#)  
[":SBUS<n>:A429:TRIGger:PATTERn:SDI" on page 741](#)  
[":SBUS<n>:A429:TRIGger:PATTERn:SSM" on page 742](#)  
[":SBUS<n>:A429:TRIGger:TYPE" on page 744](#)  
[":SBUS<n>:A429:SIGNAL" on page 736](#)  
[":SBUS<n>:A429:SPEEd" on page 738](#)  
[":SBUS<n>:A429:BASE" on page 731](#)  
[":SBUS<n>:A429:SOURce" on page 737](#)

**:SBUS<n>:A429:SIGNAl****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:A429:SIGNAl <signal>``<signal> ::= {A | B | DIFFerential}`

The :SBUS&lt;n&gt;:A429:SIGNAl command specifies the signal type:

- A – Line A (non-inverted).
- B – Line B (inverted).
- DIFFerential – Differential (A-B).

**Query Syntax**    `:SBUS<n>:A429:SIGNAl?`

The :SBUS&lt;n&gt;:A429:SIGNAl? query returns the current ARINC 429 signal type setting.

**Return Format**    `<signal><NL>``<signal> ::= {A | B | DIFF}`**Errors**    · ["-241, Hardware missing" on page 1293](#)**See Also**    · ["Introduction to :SBUS<n> Commands" on page 723](#)[":SBUS<n>:MODE" on page 727](#)[":SBUS<n>:A429:FORMAT" on page 735](#)[":SBUS<n>:A429:SPEEd" on page 738](#)[":SBUS<n>:A429:SOURce" on page 737](#)

## :SBUS<n>:A429:SOURce

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:A429:SOURce <source>

```
<source> ::= {CHANnel<n>}  
<n> ::= 1 to (# analog channels) in NR1 format
```

The :SBUS<n>:A429:SOURce command sets the source of the ARINC 429 signal.

**Query Syntax** :SBUS<n>:A429:SOURce?

The :SBUS<n>:A429:SOURce? query returns the currently set source of the ARINC 429 signal.

Use the :TRIGger:LEVel:HIGH and :TRIGger:LEVel:LOW commands to set the thresold levels for the selected source.

**Return Format** <source><NL>

**See Also**

- "[:TRIGger:LEVel:HIGH](#)" on page 1054
- "[:TRIGger:LEVel:LOW](#)" on page 1055
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:MODE](#)" on page 727
- "[:SBUS<n>:A429:TRIGger:TYPE](#)" on page 744
- "[:SBUS<n>:A429:SIGNal](#)" on page 736
- "[:SBUS<n>:A429:SPEEd](#)" on page 738
- "[:SBUS<n>:A429:FORMAT](#)" on page 735
- "[Introduction to :TRIGger Commands](#)" on page 1047

**:SBUS<n>:A429:SPEed****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:A429:SPEed <speed>`  
                  `<speed> ::= {LOW | HIGH}`

The :SBUS<n>:A429:SPEed command specifies the signal speed:

- LOW – 12.5 kb/s.
- HIGH – 100 kb/s.

**Query Syntax**    `:SBUS<n>:A429:SPEed?`

The :SBUS<n>:A429:SPEed? query returns the current ARINC 429 signal speed setting.

**Return Format**    `<speed><NL>`  
                  `<speed> ::= {LOW | HIGH}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
                • [":SBUS<n>:MODE" on page 727](#)  
                • [":SBUS<n>:A429:SIGNal" on page 736](#)  
                • [":SBUS<n>:A429:FORMAT" on page 735](#)  
                • [":SBUS<n>:A429:SOURce" on page 737](#)

## :SBUS<n>:A429:TRIGger:LABEL

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:A429:TRIGger:LABEL <value>

```
<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string>
          from 0-255 or "0xXX" (don't care)

<hex> ::= #Hnn where n ::= {0,...,9 | A,...,F}

<octal> ::= #Qnnn where n ::= {0,...,7}

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}
```

The :SBUS<n>:A429:TRIGger:LABEL command defines the ARINC 429 label value when labels are used in the selected trigger type.

To set the label value to don't cares (0xXX), set the value to -1.

**Query Syntax** :SBUS<n>:A429:TRIGger:LABEL?

The :SBUS<n>:A429:TRIGger:LABEL? query returns the current label value in decimal format.

**Return Format** <value><NL> in decimal format

**Errors**

- ["-241, Hardware missing"](#) on page 1293

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 744

## :SBUS<n>:A429:TRIGger:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:A429:TRIGger:PATTERn:DATA <string>`  
`<string> ::= "nn...n" where n ::= {0 | 1 | x}, length depends on FORMAT`

The :SBUS<n>:A429:TRIGger:PATTERn:DATA command defines the ARINC 429 data pattern resource according to the string parameter. This pattern controls the data pattern searched for in each ARINC 429 word.

### NOTE

If more bits are sent for <string> than specified by the :SBUS<n>:A429:FORMAT command, the most significant bits will be truncated.

**Query Syntax**    `:SBUS<n>:A429:TRIGger:PATTERn:DATA?`

The :SBUS<n>:A429:TRIGger:PATTERn:DATA? query returns the current settings of the specified ARINC 429 data pattern resource in the binary string format.

**Return Format**    `<string><NL> in nondecimal format`

**Errors**

- ["-241, Hardware missing"](#) on page 1293

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 744
- [":SBUS<n>:A429:TRIGger:PATTERn:SDI"](#) on page 741
- [":SBUS<n>:A429:TRIGger:PATTERn:SSM"](#) on page 742

## :SBUS<n>:A429:TRIGger:PATTERn:SDI

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:A429:TRIGger:PATTERn:SDI <string>

<string> ::= "nn" where n ::= {0 | 1 | X}, length always 2 bits

The :SBUS<n>:A429:TRIGger:PATTERn:SDI command defines the ARINC 429 two-bit SDI pattern resource according to the string parameter. This pattern controls the SDI pattern searched for in each ARINC 429 word.

The specified SDI is only used if the :SBUS<n>:A429:FORMAT includes the SDI field.

**Query Syntax** :SBUS<n>:A429:TRIGger:PATTERn:SDI?

The :SBUS<n>:A429:TRIGger:PATTERn:SDI? query returns the current settings of the specified ARINC 429 two-bit SDI pattern resource in the binary string format.

**Return Format** <string><NL> in nondecimal format

**Errors**

- ["-241, Hardware missing"](#) on page 1293

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":SBUS<n>:A429:FORMAT"](#) on page 735
- [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 744
- [":SBUS<n>:A429:TRIGger:PATTERn:DATA"](#) on page 740
- [":SBUS<n>:A429:TRIGger:PATTERn:SSM"](#) on page 742

**:SBUS<n>:A429:TRIGger:PATTERn:SSM****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:A429:TRIGger:PATTERn:SSM <string>``<string> ::= "nn" where n ::= {0 | 1 | X}, length always 2 bits`

The :SBUS<n>:A429:TRIGger:PATTERn:SSM command defines the ARINC 429 two-bit SSM pattern resource according to the string parameter. This pattern controls the SSM pattern searched for in each ARINC 429 word.

The specified SSM is only used if the :SBUS<n>:A429:FORMAT includes the SSM field.

**Query Syntax**    `:SBUS<n>:A429:TRIGger:PATTERn:SSM?`

The :SBUS<n>:A429:TRIGger:PATTERn:SSM? query returns the current settings of the specified ARINC 429 two-bit SSM pattern resource in the binary string format.

**Return Format**    `<string><NL> in nondecimal format`**Errors**

- 241, [Hardware missing](#) on page 1293

**See Also**

- [Introduction to :TRIGger Commands](#) on page 1047

- [:SBUS<n>:A429:FORMAT](#) on page 735

- [:SBUS<n>:A429:TRIGger:TYPE](#) on page 744

- [:SBUS<n>:A429:TRIGger:PATTERn:DATA](#) on page 740

- [:SBUS<n>:A429:TRIGger:PATTERn:SDI](#) on page 741

## :SBUS<n>:A429:TRIGger:RANGE

**N** (see [page 1334](#))

**Command Syntax**    :SBUS<n>:A429:TRIGger:RANGE <min>,<max>  
                   <min> ::= 8-bit integer in decimal, <hex>, <octal>, or <string>  
                   from 0-255  
                   <max> ::= 8-bit integer in decimal, <hex>, <octal>, or <string>  
                   from 0-255  
                   <hex> ::= #Hnn where n ::= {0,...,9 | A,...,F}  
                   <octal> ::= #Qnnn where n ::= {0,...,7}  
                   <string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}

The :SBUS<n>:A429:TRIGger:RANGE command defines a range of ARINC 429 label values. This range is used when the LRAnge trigger type is selected.

**Query Syntax**    :SBUS<n>:A429:TRIGger:RANGE?

The :SBUS<n>:A429:TRIGger:RANGE? query returns the current label values in decimal format.

**Return Format**    <min>,<max><NL> in decimal format

**Errors**    · ["-241, Hardware missing"](#) on page 1293

**See Also**    · ["Introduction to :TRIGger Commands"](#) on page 1047  
                   · [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 744

## :SBUS<n>:A429:TRIGger:TYPE

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:A429:TRIGger:TYPE &lt;condition&gt;</code>
	<pre>&lt;condition&gt; ::= {WSTArt   WSTOp   LABel   LBITS   PERRor   WERRor                     GERRor   WGERRrors   ALLerrors   LRANGE   ABITS                     AOBits   AZBits}</pre>

The :SBUS<n>:A429:TRIGger command sets the ARINC 429 trigger on condition:

- WSTArt – triggers on the start of a word.
- WSTOp – triggers at the end of a word.
- LABel – triggers on the specified label value.
- LBITS – triggers on the label and the other word fields as specified.
- LRANGE – triggers on a label within a min/max range.
- PERRor – triggers on words with a parity error.
- WERRor – triggers on an intra-word coding error.
- GERRor – triggers on an inter-word gap error.
- WGERRrors – triggers on either a Word or Gap Error.
- ALLerrors – triggers on any of the above errors.
- ABITS – triggers on any bit, which will therefore form an eye diagram.
- AZBits – triggers on any bit with a value of zero.
- AOBits – triggers on any bit with a value of one.

<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:A429:TRIGger:TYPE?</code>
---------------------	------------------------------------------------

The :SBUS<n>:A429:TRIGger:TYPE? query returns the current ARINC 429 trigger on condition.

<b>Return Format</b>	<code>&lt;condition&gt;&lt;NL&gt;</code>
	<pre>&lt;condition&gt; ::= {WSTA   WSTO   LAB   LBIT   PERR   WERR   GERR   WGERR                     ALL   LRAN   ABIT   AOB   AZB}</pre>

<b>Errors</b>	<a href="#">"-241, Hardware missing" on page 1293</a>
---------------	-------------------------------------------------------

<b>See Also</b>	<ul style="list-style-type: none"> <li><a href="#">"Introduction to :SBUS&lt;n&gt; Commands" on page 723</a></li> <li><a href="#">":SBUS&lt;n&gt;:MODE" on page 727</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:TRIGger:LABel" on page 739</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:TRIGger:PATTERn:DATA" on page 740</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:TRIGger:PATTERn:SDI" on page 741</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:TRIGger:PATTERn:SSM" on page 742</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:TRIGger:RANGE" on page 743</a></li> <li><a href="#">":SBUS&lt;n&gt;:A429:SOURce" on page 737</a></li> </ul>
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :SBUS<n>:CAN Commands

**NOTE**

These commands are valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

**Table 112:**:SBUS<n>:CAN Commands Summary

Command	Query	Options and Query Returns
n/a	:SBUS<n>:CAN:COUNT:ER Ror? (see <a href="#">page 748</a> )	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:OV ERload? (see <a href="#">page 749</a> )	<frame_count> ::= 0 in NR1 format
:SBUS<n>:CAN:COUNT:RE Set (see <a href="#">page 750</a> )	n/a	n/a
n/a	:SBUS<n>:CAN:COUNT:SP EC? (see <a href="#">page 751</a> )	<spec_error_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:TO Tal? (see <a href="#">page 752</a> )	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:CAN:COUNT:UT ILization? (see <a href="#">page 753</a> )	<percent> ::= floating-point in NR3 format
:SBUS<n>:CAN:DISPLAY <type> (see <a href="#">page 754</a> )	:SBUS<n>:CAN:DISPLAY? (see <a href="#">page 754</a> )	<type> ::= {HEXadecimal   SYMBOLic}
:SBUS<n>:CAN:FDSPoint <value> (see <a href="#">page 755</a> )	:SBUS<n>:CAN:FDSPoint ? (see <a href="#">page 755</a> )	<value> ::= even numbered percentages from 30 to 90 in NR3 format.
:SBUS<n>:CAN:FDSTanda rd <std> (see <a href="#">page 756</a> )	:SBUS<n>:CAN:FDSTanda rd? (see <a href="#">page 756</a> )	<std> ::= {ISO   NISO}
:SBUS<n>:CAN:SAMPLEpo int <percent> (see <a href="#">page 757</a> )	:SBUS<n>:CAN:SAMPLEpo int? (see <a href="#">page 757</a> )	<percent> ::= 30.0 to 90.0 in NR3 format
:SBUS<n>:CAN:SIGNAl:B AUDrate <baudrate> (see <a href="#">page 758</a> )	:SBUS<n>:CAN:SIGNAl:B AUDrate? (see <a href="#">page 758</a> )	<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments, or 5000000
:SBUS<n>:CAN:SIGNAl:D EFinition <value> (see <a href="#">page 759</a> )	:SBUS<n>:CAN:SIGNAl:D EFinition? (see <a href="#">page 759</a> )	<value> ::= {CANH   CANL   RX   TX   DIFFerential   DIFL   DIFH}
:SBUS<n>:CAN:SIGNAl:F DBaudrate <baudrate> (see <a href="#">page 760</a> )	:SBUS<n>:CAN:SIGNAl:F DBaudrate? (see <a href="#">page 760</a> )	<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.

**Table 112:**:SBUS< n >:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS< n >:CAN:SOURCE <source> (see page 761)	:SBUS< n >:CAN:SOURce? (see page 761)	<source> ::= {CHANnel< n >   EXTernal} for DSO models <source> ::= {CHANnel< n >   DIGItal< d >   } for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS< n >:CAN:TRIGGER <condition> (see page 762)	:SBUS< n >:CAN:TRIGger? (see page 763)	<condition> ::= {SOF   EOF   IDData   DATA   FDData   IDRemeTe   IDEither   ERRor   ACKerror   FORMerror   STUFFerror   CRCerror   SPECerror   ALLerrors   BRSBit   CRCDBit   EBActive   EBPassive   OVERload   MESSage   MSIGnal   FDMSignal}
:SBUS< n >:CAN:TRIGGER: IDFilter {{0   OFF}   {1   ON}} (see page 765)	:SBUS< n >:CAN:TRIGger: IDFilter? (see page 765)	{0   1}
:SBUS< n >:CAN:TRIGGER: PATtern:DATA <string> (see page 766)	:SBUS< n >:CAN:TRIGger: PATtern:DATA? (see page 766)	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS< n >:CAN:TRIGGER: PATtern:DATA:DLC <dlc> (see page 767)	:SBUS< n >:CAN:TRIGger: PATtern:DATA:DLC? (see page 767)	<dlc> ::= integer between -1 (don't care) and 64, in NR1 format.
:SBUS< n >:CAN:TRIGGER: PATtern:DATA:LENGTH <length> (see page 768)	:SBUS< n >:CAN:TRIGger: PATtern:DATA:LENGTH? (see page 768)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS< n >:CAN:TRIGGER: PATtern:DATA:START <start> (see page 769)	:SBUS< n >:CAN:TRIGger: PATtern:DATA:START? (see page 769)	<start> ::= integer between 0 and 63, in NR1 format.
:SBUS< n >:CAN:TRIGGER: PATtern:ID <string> (see page 770)	:SBUS< n >:CAN:TRIGger: PATtern:ID? (see page 770)	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS< n >:CAN:TRIGGER: PATtern:ID:MODE <value> (see page 771)	:SBUS< n >:CAN:TRIGger: PATtern:ID:MODE? (see page 771)	<value> ::= {STANDARD   EXTENDED}

**Table 112:**:SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:TRIGger: SYMBolic:MESSage <name> (see page 772)	:SBUS<n>:CAN:TRIGger: SYMBolic:MESSage? (see page 772)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGGER: SYMBolic:SIGNAl <name> (see page 773)	:SBUS<n>:CAN:TRIGGER: SYMBolic:SIGNAl? (see page 773)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger: SYMBolic:VALue <data> (see page 774)	:SBUS<n>:CAN:TRIGger: SYMBolic:VALue? (see page 774)	<data> ::= value in NR3 format

:SBUS<n>:CAN:COUNT:ERRor

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:CAN:COUNT:ERRor?

Returns the error frame count.

**Return Format** <frame\_count><NL>

<frame\_count> ::= integer in NR1 format

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:CAN:COUNT:RESet" on page 750](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

**:SBUS<n>:CAN:COUNT:OVERload**

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:CAN:COUNT:OVERload?

Returns the overload frame count.

**Return Format** <frame\_count><NL>

<frame\_count> ::= 0 in NR1 format

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:CAN:COUNT:RESET" on page 750](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

:SBUS<n>:CAN:COUNT:RESet

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:CAN:COUNT:RESet

Resets the frame counters.

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:CAN:COUNT:ERRor" on page 748](#)  
• [":SBUS<n>:CAN:COUNT:OVERload" on page 749](#)  
• [":SBUS<n>:CAN:COUNT:TOTal" on page 752](#)  
• [":SBUS<n>:CAN:COUNt:UTILization" on page 753](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

**:SBUS<n>:CAN:COUNT:SPEC**

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:CAN:COUNT:SPEC?

Returns the Spec error (Ack + Form + Stuff + CRC errors) count.

**Return Format** <spec\_error\_count><NL>

<spec\_error\_count> ::= integer in NR1 format

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:CAN:COUNT:RESET" on page 750](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

**:SBUS<n>:CAN:COUNT:TOTal****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:CAN:COUNT:TOTal?`

Returns the total frame count.

**Return Format**    `<frame_count><NL>``<frame_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:CAN:COUNT:RESet" on page 750](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

**:SBUS<n>:CAN:COUNT:UTILization**

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:CAN:COUNT:UTILization?

Returns the percent utilization.

**Return Format** <percent><NL>

<percent> ::= floating-point in NR3 format

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:CAN:COUNT:RESet" on page 750](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:CAN Commands" on page 745](#)

**:SBUS<n>:CAN:DISPlay****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:DISPlay <type>`  
                  `<type> ::= {HEXadecimal | SYMBolic}`

The :SBUS<n>:CAN:DISPlay command specifies, when CAN symbolic data is loaded into the oscilloscope, whether symbolic values (from the DBC file) or hexadecimal values are displayed in the decode waveform and the Lister window.

**Query Syntax**    `:SBUS<n>:CAN:DISPlay?`

The :SBUS<n>:CAN:DISPlay? query returns the CAN decode display type.

**Return Format**    `<type><NL>`  
                  `<type> ::= {HEX | SYMB}`

**See Also**    • [":RECall:DBC\[:STARt\]" on page 686](#)

**:SBUS<n>:CAN:FDSPoint****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:CAN:FDSPoint <value>`

`<value>` ::= even numbered percentages from 30 to 90 in NR3 format.

The :SBUS<n>:CAN:FDSPoint command sets the point during the bit time where the bit level is sampled to determine whether the bit is dominant or recessive. The sample point represents the percentage of time between the beginning of the bit time to the end of the bit time.

**Query Syntax**    `:SBUS<n>:CAN:FDSPoint?`

The :SBUS<n>:CAN:FDSPoint? query returns the current CAN FD sample point setting.

**Return Format**    `<value><NL>`

`<value>` ::= even numbered percentages from 30 to 90 in NR3 format.

**See Also**    · [":SBUS<n>:CAN:SIGNal:FDBaudrate"](#) on page 760

**:SBUS<n>:CAN:FDSTandard****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:FDSTandard <std>`  
                  `<std> ::= {ISO | NISO}`

The :SBUS<n>:CAN:FDSTandard command lets you pick the standard that will be used when decoding or triggering on FD frames, ISO, or non-ISO.

This setting has no effect on the processing of non-FD (classical) frames.

**Query Syntax**    `:SBUS<n>:CAN:FDSTandard?`

The :SBUS<n>:CAN:FDSTandard? query returns the selected CAN FD frame decode standard.

**Return Format**    `<std>`  
                  `<std> ::= {ISO | NISO}`

**See Also**    • [":SBUS<n>:CAN:FDSPoint"](#) on page 755

## :SBUS<n>:CAN:SAMPLEpoint

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:SAMPLEpoint <percent>`

`<percent><NL>`

`<percent> ::= 30.0 to 90.0 in NR3 format`

The :SBUS<n>:CAN:SAMPLEpoint command sets the point during the bit time where the bit level is sampled to determine whether the bit is dominant or recessive. The sample point represents the percentage of time between the beginning of the bit time to the end of the bit time.

**Query Syntax**    `:SBUS<n>:CAN:SAMPLEpoint?`

The :SBUS<n>:CAN:SAMPLEpoint? query returns the current CAN sample point setting.

**Return Format**    `<percent><NL>`

`<percent> ::= 30.0 to 90.0 in NR3 format`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:MODE](#)" on page 727
- "[":SBUS<n>:CAN:TRIGger](#)" on page 762

## :SBUS<n>:CAN:SIGNAl:BAUDrate

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:CAN:SIGNAl:BAUDrate <baudrate>
<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments,
               or 5000000
```

The :SBUS<n>:CAN:SIGNAl:BAUDrate command sets the standard baud rate of the CAN signal from 10 kb/s to 4 Mb/s in 100 b/s increments. If you enter a baud rate that is not divisible by 100 b/s, the baud rate is set to the nearest baud rate divisible by 100 b/s.

You can also set the baud rate of the CAN signal to 5 Mb/s. Fractional baud rates between 4 Mb/s and 5 Mb/s are not allowed.

If the baud rate you select does not match the system baud rate, false triggers may occur.

**Query Syntax**

```
:SBUS<n>:CAN:SIGNAl:BAUDrate?
```

The :SBUS<n>:CAN:SIGNAl:BAUDrate? query returns the current CAN baud rate setting.

**Return Format**

```
<baudrate><NL>
<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments,
               or 5000000
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:MODE](#)" on page 727
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:SIGNAl:DEFinition](#)" on page 759
- "[:SBUS<n>:CAN:SOURce](#)" on page 761

## :SBUS<n>:CAN:SIGNAl:DEFinition

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:SIGNAl:DEFinition <value>`  
`<value> ::= {CANH | CANL | RX | TX | DIFFerential | DIFL | DIFH}`

The :SBUS<n>:CAN:SIGNAl:DEFinition command sets the CAN signal type when :SBUS<n>:CAN:TRIGger is set to SOF (start of frame). These signals can be set to:

Dominant high signals:

- CANH – the actual CAN\_H differential bus signal.
- DIFH – the CAN differential (H-L) bus signal connected to an analog source channel using a differential probe.

Dominant low signals:

- CANL – the actual CAN\_L differential bus signal.
- RX – the Receive signal from the CAN bus transceiver.
- TX – the Transmit signal to the CAN bus transceiver.
- DIFL – the CAN differential (L-H) bus signal connected to an analog source channel using a differential probe.
- DIFFerential – the CAN differential bus signal connected to an analog source channel using a differential probe. This is the same as DIFL.

**Query Syntax**    `:SBUS<n>:CAN:SIGNAl:DEFinition?`

The :SBUS<n>:CAN:SIGNAl:DEFinition? query returns the current CAN signal type.

**Return Format**    `<value><NL>`  
`<value> ::= {CANH | CANL | RX | TX | DIFL | DIFH}`

**See Also**    ["Introduction to :TRIGger Commands" on page 1047](#)  
[":SBUS<n>:MODE" on page 727](#)  
[":SBUS<n>:CAN:SIGNAl:BAUDrate" on page 758](#)  
[":SBUS<n>:CAN:SOURce" on page 761](#)  
[":SBUS<n>:CAN:TRIGger" on page 762](#)

**:SBUS<n>:CAN:SIGNAl:FDBaudrate****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:SIGNAl:FDBaudrate <baudrate>`  
`<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.`

The :SBUS<n>:CAN:SIGNAl:FDBaudrate command sets the CAN FD baud rate from 10 kb/s to 10 Mb/s in 100 b/s increments. If you enter a baud rate that is not divisible by 100 b/s, the baud rate is set to the nearest baud rate divisible by 100 b/s.

For CAN FD, both the standard rate settings (see :SBUS<n>:CAN:SIGNAl:BAUDrate) and the FD rate settings must be set correctly; otherwise, false triggers may occur.

**Query Syntax**    `:SBUS<n>:CAN:SIGNAl:FDBaudrate?`

The :SBUS<n>:CAN:SIGNAl:FDBaudrate? query returns the current CAN FD baud rate setting.

**Return Format**    `<baudrate><NL>`  
`<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.`

**See Also**

- "[:SBUS<n>:CAN:FDSPoint](#)" on page 755
- "[:SBUS<n>:CAN:SIGNAl:BAUDrate](#)" on page 758

## :SBUS<n>:CAN:SOURce

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:SOURce <source>`

```
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:CAN:SOURce command sets the source for the CAN signal.

**Query Syntax**    `:SBUS<n>:CAN:SOURce?`

The :SBUS<n>:CAN:SOURce? query returns the current source for the CAN signal.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:MODE](#)" on page 727
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:SIGNal:DEFinition](#)" on page 759

## :SBUS<n>:CAN:TRIGger

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:CAN:TRIGger <condition>

```
<condition> ::= {SOF | EOF | IDData | DATA | FDData | IDRemote
| IDEEither | ERRor | ACKerror | FORMrror | STUFFrror | CRCrror
| SPECrror | ALLerrors | BRSBIT | CRCDbit | EBActive | EBPassive
| OVERload | MESSage | MSIGnal | FDMSignal}
```

The :SBUS<n>:CAN:TRIGger command sets the CAN trigger on condition:

Condition	Front-panel name	Description	Filter by ID*
SOF	SOF - Start of Frame	Triggers at the start bit for both data and overload frames.	
EOF	EOF - End of Frame	Triggers at the end of any frame.	X
IDEither	Frame ID	Triggers on any standard CAN (data or remote) or CAN FD frame at the end of the 11- or 29-bit ID field.	
IDData	Data Frame ID (non-FD)	Triggers on standard CAN data frames at the end of the 11- or 29-bit ID field.	
FDData	Data Frame ID and Data (non-FD)	Triggers on any standard CAN data frame at the end of the last data byte defined in the trigger. The DLC of the packet must match the number of bytes specified.	
IDRemote	Remote Frame ID	Triggers on standard CAN remote frames at the end of the 11- or 29-bit ID field.	
ERRor	Error Frame	Triggers after 6 consecutive 0s while in a data frame, at the EOF.	X
ACKerror	Acknowledge Error	Triggers on the acknowledge bit if the polarity is incorrect.	X
FORMrror	Form Error	Triggers on reserved bit errors.	X
STUFFrror	Stuff Error	Triggers on 6 consecutive 1s or 6 consecutive 0s, while in a non-error or non overload frame.	X

Condition	Front-panel name	Description	Filter by ID*
CRCerror	CRC Field Error	Triggers when the calculated CRC does not match the transmitted CRC. In addition, for FD frames, will also trigger if the Stuff Count is in error.	X
SPECerror	Spec Error (Ack or Form or Stuff or CRC)	Triggers on Ack, Form, Stuff, or CRC errors.	X
ALLerrors	All Errors	Triggers on all Spec errors and error frames.	X
BRSBIt	BRS Bit (FD)	Triggers on the BRS bit of CAN FD frames.	X
CRCDbit	CRC Delimiter Bit (FD)	Triggers on the CRC delimiter bit in CAN FD frames.	X
EBAActive	ESI Bit Active (FD)	Triggers on the ESI bit if set active.	X
EBPassive	ESI Bit Passive (FD)	Triggers on the ESI bit if set passive.	X
OVERload	Overload Frame	Triggers on an overload frame.	
MESSage	Message	Triggers on a symbolic message.	
MSIGnal	Message and Signal (non-FD)	Triggers on a symbolic message and a signal value.	
FDMSignal	Message and Signal (FD, first 8 bytes only)	Triggers on a symbolic message and a signal value, limited to the first 8 bytes of FD data.	

\* Filtering by CAN IDs is available for these trigger conditions (see :SBUS<n>:CAN:TRIGger:IDFilter).

CAN Id specification is set by the :SBUS<n>:CAN:TRIGger:PATTERn:ID and :SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE commands.

CAN Data specification is set by the :SBUS<n>:CAN:TRIGger:PATTERn:DATA command.

CAN Data Length Code is set by the :SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth command.

**Query Syntax** :SBUS<n>:CAN:TRIGger?

The :SBUS<n>:CAN:TRIGger? query returns the current CAN trigger on condition.

**Return Format** <condition><NL>

```
<condition> ::= { SOF | EOF | IDD | DATA | FDD | IDR | IDE | ERR | ACK
    | FORM | STUF | CRC | SPEC | ALL | BRSB | CRCD | EBA | EBP | OVER
    | MESS | MSIG | FDMS }
```

**Errors** • "-241, Hardware missing" on page 1293

**See Also** • "Introduction to :SBUS<n> Commands" on page 723

- "[:SBUS<n>:MODE](#)" on page 727
- "[:SBUS<n>:CAN:TRIGger:PATTERn:DATA](#)" on page 766
- "[:SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth](#)" on page 768
- "[:SBUS<n>:CAN:TRIGger:PATTERn:ID](#)" on page 770
- "[:SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE](#)" on page 771
- "[:SBUS<n>:CAN:TRIGger:IDFilter](#)" on page 765
- "[:SBUS<n>:CAN:SIGNAL:DEFinition](#)" on page 759
- "[:SBUS<n>:CAN:SOURce](#)" on page 761
- "[:RECall:DBC\[:START\]](#)" on page 686
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge](#)" on page 772
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:SIGNAl](#)" on page 773
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue](#)" on page 774

## :SBUS<n>:CAN:TRIGger:IDFilter

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:CAN:TRIGger:IDFilter {{0 | OFF} | {1 | ON}}

The :SBUS<n>:CAN:TRIGger:IDFilter command specifies, in certain error and bit trigger modes, whether triggers are filtered by CAN IDs.

**Query Syntax** :SBUS<n>:CAN:TRIGger:IDFilter?

The :SBUS<n>:CAN:TRIGger:IDFilter? query returns the CAN trigger ID filter setting.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

**See Also** • [":SBUS<n>:CAN:TRIGger"](#) on page 762

## :SBUS<n>:CAN:TRIGger:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA <string>`

```
<string> ::= "nn...n" where n ::= {0 | 1 | x | $}
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $}
```

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA command defines the CAN data pattern resource according to the string parameter. This pattern, along with the data length (set by the :SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth command), control the data pattern searched for in each CAN message.

If the string parameter starts with "0x", it is a hexadecimal string made up of hexadecimal and X (don't care) characters; otherwise, it is a binary string made up of 0, 1, and X (don't care) characters.

### NOTE

If more bits are sent for <string> than specified by the :SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth command, the most significant bits will be truncated. If the data length is changed after the <string> is programmed, the added or deleted bits will be added to or deleted from the least significant bits.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA?`

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA? query returns the current settings of the specified CAN data pattern resource in the binary string format.

**Return Format**    `<string><NL>` in nondecimal format

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :TRIGger Commands" on page 1047](#)  
                 • [":SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth" on page 768](#)  
                 • [":SBUS<n>:CAN:TRIGger:PATTERn:ID" on page 770](#)

**:SBUS<n>:CAN:TRIGger:PATTERn:DATA:DLC**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA:DLC <dLC>`

`<dLC> ::= integer between -1 (don't care) and 64, in NR1 format.`

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA:DLC command specifies the DLC value to be used in the CAN FD data trigger mode. A specific valid FD value can be specified, or -1 can be specified to indicate "don't care".

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA:START?`

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA:DLC? query returns the currently set DLC value.

**Return Format**    `<dLC><NL>`

`<dLC> ::= integer between -1 (don't care) and 64, in NR1 format.`

**See Also**    · [":SBUS<n>:CAN:TRIGger:PATTERn:DATA"](#) on page 766

**:SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth <length>`    `<length> ::= integer from 1 to 8 in NR1 format`

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth command sets the number of 8-bit bytes in the CAN data string. The number of bytes in the string can be anywhere from 1 bytes to 8 bytes (64 bits). The value for these bytes is set by the :SBUS<n>:CAN:TRIGger:PATTERn:DATA command.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth?`

The :SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth? query returns the current CAN data pattern length setting.

**Return Format**    `<count><NL>`    `<count> ::= integer from 1 to 8 in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • ["Introduction to :TRIGger Commands" on page 1047](#)  
    • [":SBUS<n>:CAN:TRIGger:PATTERn:DATA" on page 766](#)  
    • [":SBUS<n>:CAN:SOURce" on page 761](#)

## :SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt <start>

<start> ::= integer between 0 and 63, in NR1 format.

The :SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt command specifies the starting byte position for CAN FD data triggers.

CAN FD frames can have up to 64 bytes of data. You can trigger on up to 8 bytes of data. The starting byte position setting lets you trigger on data anywhere within the frame.

**Query Syntax** :SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt?

The :SBUS<n>:CAN:TRIGger:PATTern:DATA:STARt? query returns the starting byte position setting.

**Return Format** <start><NL>

<start> ::= integer between 0 and 63, in NR1 format.

**See Also**

- "[:SBUS<n>:CAN:TRIGger:PATTern:DATA](#)" on page 766
- "[:SBUS<n>:CAN:TRIGger:PATTern:DATA:LENGth](#)" on page 768

## :SBUS<n>:CAN:TRIGger:PATTern:ID

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:PATTern:ID <string>`

```
<string> ::= "nn...n" where n ::= {0 | 1 | x | $}
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $}
```

The :SBUS<n>:CAN:TRIGger:PATTern:ID command defines the CAN identifier pattern resource according to the string parameter. This pattern, along with the identifier mode (set by the :SBUS<n>:CAN:TRIGger:PATTern:ID:MODE command), control the identifier pattern searched for in each CAN message.

If the string parameter starts with "0x", it is a hexadecimal string made up of hexadecimal and X (don't care) characters; otherwise, it is a binary string made up of 0, 1, and X (don't care) characters.

### NOTE

The ID pattern resource string is always 29 bits. Only 11 of these bits are used when the :SBUS<n>:CAN:TRIGger:PATTern:ID:MODE is STANDARD.

A string longer than 29 bits is truncated to 29 bits when setting the ID pattern resource.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:PATTern:ID?`

The :SBUS<n>:CAN:TRIGger:PATTern:ID? query returns the current settings of the specified CAN identifier pattern resource in the 29-bit binary string format.

**Return Format**    `<string><NL>` in 29-bit binary string format

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :TRIGger Commands" on page 1047](#)  
                 • [":SBUS<n>:CAN:TRIGger:PATTern:ID:MODE" on page 771](#)  
                 • [":SBUS<n>:CAN:TRIGger:PATTern:DATA" on page 766](#)

## :SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE <value>`  
`<value> ::= {STANDARD | EXTENDED}`

The :SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE command sets the CAN identifier mode. STANDARD selects the standard 11-bit identifier. EXTENDED selects the extended 29-bit identifier. The CAN identifier is set by the :SBUS<n>:CAN:TRIGger:PATTERn:ID command.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE?`

The :SBUS<n>:CAN:TRIGger:PATTERn:ID:MODE? query returns the current setting of the CAN identifier mode.

**Return Format**    `<value><NL>`  
`<value> ::= {STAN | EXT}`

**Errors**    • ["-241, Hardware missing"](#) on page 1293

**See Also**    • ["Introduction to :TRIGger Commands"](#) on page 1047  
• [":SBUS<n>:MODE"](#) on page 727  
• [":SBUS<n>:CAN:TRIGger:PATTERn:DATA"](#) on page 766  
• [":SBUS<n>:CAN:TRIGger:PATTERn:DATA:LENGth"](#) on page 768  
• [":SBUS<n>:CAN:TRIGger:PATTERn:ID"](#) on page 770

**:SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge <name>`  
                  `<name> ::= quoted ASCII string`

The :SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge command specifies the message to trigger on when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN trigger mode is set to MESSage or MSIGnal.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge?`

The :SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge? query returns the specified message.

**Return Format**    `<name><NL>`  
                  `<name> ::= quoted ASCII string`

**See Also**

- "[:RECall:DBC\[:STARt\]](#)" on page 686
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl](#)" on page 773
- "[:SBUS<n>:CAN:TRIGger:SYMBolic:VALue](#)" on page 774

## :SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl <name>`  
`<name> ::= quoted ASCII string`

The :SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl command specifies the signal to trigger on when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN trigger mode is set to MSIGnAl.

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl?`

The :SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAl? query returns the specified signal.

**Return Format**    `<name><NL>`  
`<name> ::= quoted ASCII string`

**See Also**

- "[:RECall:DBC\[:STARt\]](#)" on page 686
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:TRIGger:SYMBolic:MESSAge](#)" on page 772
- "[:SBUS<n>:CAN:TRIGger:SYMBolic:VALue](#)" on page 774

## :SBUS<n>:CAN:TRIGger:SYMBOLic:VALue

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue <data>`  
`<data> ::= value in NR3 format`

The :SBUS<n>:CAN:TRIGger:SYMBOLic:VALue command specifies the signal value to trigger on when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN trigger mode is set to MSIGnAl.

**NOTE** Encoded signal values are not supported in the remote interface (even though they can be used in the front panel graphical interface).

**Query Syntax**    `:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue?`

The :SBUS<n>:CAN:TRIGger:SYMBOLic:VALue? query returns the specified signal value.

**Return Format**    `<data><NL>`

`<data> ::= value in NR3 format`

**See Also**

- "[:RECall:DBC\[:STARt\]](#)" on page 686
- "[:SBUS<n>:CAN:TRIGger](#)" on page 762
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge](#)" on page 772
- "[:SBUS<n>:CAN:TRIGger:SYMBOLic:SIGNAl](#)" on page 773

## :SBUS<n>:FLEXray Commands

**NOTE**

These commands are only valid when the FLEXray triggering and serial decode option (Option FLEX) has been licensed.

**Table 113:**:SBUS<n>:FLEXray Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:FLEXray:AUTOsetup (see page 777)	n/a	n/a
:SBUS<n>:FLEXray:BAUDrate <baudrate> (see page 778)	:SBUS<n>:FLEXray:BAUDrate? (see page 778)	<baudrate> ::= {2500000   5000000   10000000}
:SBUS<n>:FLEXray:CHANNEL <channel> (see page 779)	:SBUS<n>:FLEXray:CHANNEL? (see page 779)	<channel> ::= {A   B}
n/a	:SBUS<n>:FLEXray:COUNT:NULL? (see page 780)	<frame_count> ::= integer in NR1 format
:SBUS<n>:FLEXray:COUNT:RESET (see page 781)	n/a	n/a
n/a	:SBUS<n>:FLEXray:COUNT:SYNC? (see page 782)	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:FLEXray:COUNT:TOTAl? (see page 783)	<frame_count> ::= integer in NR1 format
:SBUS<n>:FLEXray:SOURCE <source> (see page 784)	:SBUS<n>:FLEXray:SOURCE? (see page 784)	<source> ::= {CHANNEL<n>} <n> ::= 1-2 or 1-4 in NR1 format
:SBUS<n>:FLEXray:TRIGGER <condition> (see page 785)	:SBUS<n>:FLEXray:TRIGGER? (see page 785)	<condition> ::= {FRAME   ERROR   EVENT}
:SBUS<n>:FLEXray:TRIGGER:ERROR:TYPE <error_type> (see page 786)	:SBUS<n>:FLEXray:TRIGGER:ERROR:TYPE? (see page 786)	<error_type> ::= {ALL   HCRC   FCRC}
:SBUS<n>:FLEXray:TRIGGER:EVENT:AUTOSET (see page 787)	n/a	n/a
:SBUS<n>:FLEXray:TRIGGER:EVENT:BSS:ID <frame_id> (see page 788)	:SBUS<n>:FLEXray:TRIGGER:EVENT:BSS:ID? (see page 788)	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047

**Table 113:**:SBUS<n>:FLEXray Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:FLEXray:TRIG ger:EVENT:TYPE <event> (see <a href="#">page 789</a> )	:SBUS<n>:FLEXray:TRIG ger:EVENT:TYPE? (see <a href="#">page 789</a> )	<event> ::= {WAKEup   TSS   {FES   DTS}   BSS}
:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCBase <cycle_count_base> (see <a href="#">page 790</a> )	:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCBase? (see <a href="#">page 790</a> )	<cycle_count_base> ::= integer from 0-63
:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCRepetitio n <cycle_count_repetiti on> (see <a href="#">page 791</a> )	:SBUS<n>:FLEXray:TRIG ger:FRAMe:CCRepetitio n? (see <a href="#">page 791</a> )	<cycle_count_repetition> ::= {ALL   <rep #>} <rep #> ::= integer values 2, 4, 8, 16, 32, or 64
:SBUS<n>:FLEXray:TRIG ger:FRAMe:ID <frame_id> (see <a href="#">page 792</a> )	:SBUS<n>:FLEXray:TRIG ger:FRAMe:ID? (see <a href="#">page 792</a> )	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047
:SBUS<n>:FLEXray:TRIG ger:FRAMe:TYPE <frame_type> (see <a href="#">page 793</a> )	:SBUS<n>:FLEXray:TRIG ger:FRAMe:TYPE? (see <a href="#">page 793</a> )	<frame_type> ::= {NORMAl   STARtup   NULL   SYNC   NSTArtup   NNULL   NSYNC   ALL}

## :SBUS< n >:FLEXray:AUTosetup

**N** (see [page 1334](#))

**Command Syntax** :SBUS< n >:FLEXray:AUTosetup

The :SBUS< n >:FLEXray:AUTosetup command automatically configures oscilloscope settings to facilitate FlexRay triggering and serial decode.

- Sets the selected source channel's impedance to 50 Ohms.
- Sets the selected source channel's probe attenuation to 10:1.
- Sets the trigger level (on the selected source channel) to -300 mV.
- Turns on trigger Noise Reject.
- Turns on Serial Decode.
- Sets the trigger to the specified serial bus (n of SBUS< n >).

**See Also** ["Introduction to :TRIGger Commands" on page 1047](#)  
[":SBUS< n >:FLEXray:TRIGger" on page 785](#)  
[":SBUS< n >:FLEXray:BAUDrate" on page 778](#)  
[":TRIGger\[:EDGE\]:LEVel" on page 1073](#)  
[":SBUS< n >:FLEXray:SOURce" on page 784](#)

**:SBUS<n>:FLEXray:BAUDrate****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:FLEXray:BAUDrate <baudrate>``<baudrate> ::= {2500000 | 5000000 | 10000000}`

The :SBUS<n>:FLEXray:BAUDrate command specifies the baud rate as 2.5 Mb/s, 5 Mb/s, or 10 Mb/s.

**Query Syntax**    `:SBUS<n>:FLEXray:BAUDrate?`

The :SBUS<n>:FLEXray:BAUDrate? query returns the current baud rate setting.

**Return Format**    `<baudrate><NL>``<baudrate> ::= {2500000 | 5000000 | 10000000}`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- "[:SBUS<n>:FLEXray Commands](#)" on page 775

**:SBUS<n>:FLEXray:CHANnel****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:CHANnel <channel>`  
                      `<channel> ::= {A | B}`

The :SBUS<n>:FLEXray:CHANnel command specifies the bus channel, A or B, of the FlexRay signal.

**Query Syntax**    `:SBUS<n>:FLEXray:CHANnel?`

The :SBUS<n>:FLEXray:CHANnel? query returns the current bus channel setting.

**Return Format**    `<channel><NL>`  
                      `<channel> ::= {A | B}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:FLEXray Commands](#)" on page 775

**:SBUS<n>:FLEXray:COUNT:NULL****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:FLEXray:COUNT:NULL?`

Returns the FlexRay null frame count.

**Return Format**    `<frame_count><NL>``<frame_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:FLEXray:COUNT:RESet" on page 781](#)[":SBUS<n>:FLEXray:COUNT:TOTal" on page 783](#)[":SBUS<n>:FLEXray:COUNT:SYNC" on page 782](#)["Introduction to :SBUS<n> Commands" on page 723](#)[":SBUS<n>:MODE" on page 727](#)[":SBUS<n>:FLEXray Commands" on page 775](#)

**:SBUS<n>:FLEXray:COUNT:RESet**

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:FLEXray:COUNT:RESet

Resets the FlexRay frame counters.

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:FLEXray:COUNT:NULL" on page 780](#)

• [":SBUS<n>:FLEXray:COUNT:TOTal" on page 783](#)

• [":SBUS<n>:FLEXray:COUNT:SYNC" on page 782](#)

• ["Introduction to :SBUS<n> Commands" on page 723](#)

• [":SBUS<n>:MODE" on page 727](#)

• [":SBUS<n>:FLEXray Commands" on page 775](#)

**:SBUS<n>:FLEXray:COUNT:SYNC****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:FLEXray:COUNT:SYNC?`

Returns the FlexRay sync frame count.

**Return Format**    `<frame_count><NL>``<frame_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:FLEXray:COUNT:RESet" on page 781](#)[":SBUS<n>:FLEXray:COUNT:TOTal" on page 783](#)[":SBUS<n>:FLEXray:COUNT:NULL" on page 780](#)["Introduction to :SBUS<n> Commands" on page 723](#)[":SBUS<n>:MODE" on page 727](#)[":SBUS<n>:FLEXray Commands" on page 775](#)

**:SBUS<n>:FLEXray:COUNT:TOTal****N**(see [page 1334](#))**Query Syntax**    `:SBUS<n>:FLEXray:COUNT:TOTal?`

Returns the FlexRay total frame count.

**Return Format**    `<frame_count><NL>``<frame_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:FLEXray:COUNT:RESet" on page 781](#)  
• [":SBUS<n>:FLEXray:COUNT:TOTal" on page 783](#)  
• [":SBUS<n>:FLEXray:COUNT:NULL" on page 780](#)  
• [":SBUS<n>:FLEXray:COUNT:SYNC" on page 782](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:FLEXray Commands" on page 775](#)

**:SBUS<n>:FLEXray:SOURce****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:SOURce <source>`  
`<source> ::= {CHANnel<n>}`  
`<n> ::= {1 | 2 | 3 | 4}`

The :SBUS<n>:FLEXray:SOURce command specifies the input source for the FlexRay signal.

**Query Syntax**    `:SBUS<n>:FLEXray:SOURce?`

The :SBUS<n>:FLEXray:SOURce? query returns the current source for the FlexRay signal.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785
- "[:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE](#)" on page 789
- "[:SBUS<n>:FLEXray:AUTosetup](#)" on page 777

## :SBUS<n>:FLEXray:TRIGger

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:TRIGger <condition>`  
`<condition> ::= {FRAMe | ERRor | EVENT}`

The :SBUS<n>:FLEXray:TRIGger command sets the FLEXray trigger on condition:

- FRAMe – triggers on specified frames (without errors).
- ERRor – triggers on selected active error frames and unknown bus conditions.
- EVENT – triggers on specified FlexRay event/symbol.

**Query Syntax**    `:SBUS<n>:FLEXray:TRIGger?`

The :SBUS<n>:FLEXray:TRIGger? query returns the current FLEXray trigger on condition.

**Return Format**    `<condition><NL>`  
`<condition> ::= {FRAM | ERR | EVEN}`

**See Also**    ["Introduction to :TRIGger Commands" on page 1047](#)  
[":TRIGger:MODE" on page 1056](#)  
[":SBUS<n>:FLEXray:TRIGger:ERRor:TYPE" on page 786](#)  
[":SBUS<n>:FLEXray:TRIGger:EVENT:AUToset" on page 787](#)  
[":SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID" on page 788](#)  
[":SBUS<n>:FLEXray:TRIGger:EVENT:TYPE" on page 789](#)  
[":SBUS<n>:FLEXray:TRIGger:FRAME:CCBase" on page 790](#)  
[":SBUS<n>:FLEXray:TRIGger:FRAME:CCRepetition" on page 791](#)  
[":SBUS<n>:FLEXray:TRIGger:FRAME:ID" on page 792](#)  
[":SBUS<n>:FLEXray:TRIGger:FRAME:TYPE" on page 793](#)

**:SBUS<n>:FLEXray:TRIGger:ERRor:TYPE****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:TRIGger:ERRor:TYPE <error_type>`  
`<error_type> ::= {ALL | HCRC | FCRC}`

Selects the FlexRay error type to trigger on. The error type setting is only valid when the FlexRay trigger mode is set to ERRor.

- ALL – triggers on ALL errors.
- HCRC – triggers on only Header CRC errors.
- FCRC – triggers on only Frame CRC errors.

**Query Syntax**    `:SBUS<n>:FLEXray:TRIGger:ERRor:TYPE?`

The `:SBUS<n>:FLEXray:TRIGger:ERRor:TYPE?` query returns the currently selected FLEXray error type.

**Return Format**    `<error_type><NL>`  
`<error_type> ::= {ALL | HCRC | FCRC}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785

**:SBUS<n>:FLEXray:TRIGger:EVENT:AUToSet**

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:FLEXray:TRIGger:EVENT:AUToSet

The :SBUS<n>:FLEXray:TRIGger:EVENT:AUToSet command automatically configures oscilloscope settings (as shown on the display) for the selected event trigger.

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE](#)" on page 789
- "[:SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID](#)" on page 788
- "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785
- "[:SBUS<n>:FLEXray:BAUDrate](#)" on page 778
- "[:TRIGger\[:EDGE\]:LEVel](#)" on page 1073
- "[:SBUS<n>:FLEXray:SOURce](#)" on page 784

## :SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID <frame_id>
  <frame_id> ::= {ALL | <frame #>}
  <frame #> ::= integer from 1-2047
```

The :SBUS<N>:FLEXray:TRIGger:EVENT:BSS:ID command sets the frame ID used by the Byte Start Sequence (BSS) event trigger. This setting is only valid if the trigger mode is EVENT and the EVENT:TYPE is BSS.

**Query Syntax**

```
:SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID?
```

The :SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID? query returns the current frame ID setting for the Byte Start Sequence (BSS) event trigger setup.

**Return Format**

```
<frame_id><NL>
  <frame_id> ::= {ALL | <frame #>}
  <frame #> ::= integer from 1-2047
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:FLEXray:TRIGger:EVENT:TYPE](#)" on page 789
- "[":SBUS<n>:FLEXray:TRIGger:EVENT:AUToset](#)" on page 787
- "[":TRIGger:MODE](#)" on page 1056
- "[":SBUS<n>:FLEXray:TRIGger](#)" on page 785

## :SBUS<n>:FLEXray:TRIGger:EVENT:TYPE

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE <event>`  
`<event> ::= {WAKEup | TSS | {FES | DTS} | BSS}`

Selects the FlexRay event to trigger on. The event setting is only valid when the FlexRay trigger mode is set to EVENT.

- WAKEup – triggers on Wake-Up event.
- TSS – triggers on Transmission Start Sequence event.
- FES – triggers on either Frame End or Dynamic Trailing Sequence event.
- DTS – triggers on either Frame End or Dynamic Trailing Sequence event.
- BSS – triggers on Byte Start Sequence event.

**Query Syntax**    `:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE?`

The `:SBUS<n>:FLEXray:TRIGger:EVENT:TYPE?` query returns the currently selected FLEXray event.

**Return Format**    `<event><NL>`  
`<event> ::= {WAK | TSS | {FES | DTS} | BSS}`

**See Also**    ["Introduction to :TRIGger Commands" on page 1047](#)  
[":SBUS<n>:FLEXray:TRIGger:EVENT:AUToset" on page 787](#)  
[":SBUS<n>:FLEXray:TRIGger:EVENT:BSS:ID" on page 788](#)  
[":SBUS<n>:FLEXray:TRIGger" on page 785](#)  
[":SBUS<n>:FLEXray:AUTosetup" on page 777](#)  
[":SBUS<n>:FLEXray:SOURce" on page 784](#)

**:SBUS<n>:FLEXray:TRIGger:FRAMe:CCBase****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:TRIGger:FRAMe:CCBase <cycle_count_base>`  
`<cycle_count_base> ::= integer from 0-63`

The :SBUS<n>:FLEXray:TRIGger:FRAMe:CCBase command sets the base of the FlexRay cycle count (in the frame header) to trigger on. The cycle count base setting is only valid when the FlexRay trigger mode is set to FRAME.

**Query Syntax**    `:SBUS<n>:FLEXray:TRIGger:FRAMe:CCBase?`

The :SBUS<n>:FLEXray:TRIGger:FRAMe:CCBase? query returns the current cycle count base setting for the FlexRay frame trigger setup.

**Return Format**    `<cycle_count_base><NL>`  
`<cycle_count_base> ::= integer from 0-63`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785

## :SBUS<n>:FLEXray:TRIGger:FRAMe:CCRepetition

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:FLEXray:TRIGger:FRAMe:CCRepetition <cycle_count_repetition>
<cycle_count_repetition> ::= {ALL | <rep #>}
<rep #> ::= integer values 2, 4, 8, 16, 32, or 64
```

The :SBUS<n>:FLEXray:TRIGger:FRAMe:CCRepetition command sets the repetition number of the FlexRay cycle count (in the frame header) to trigger on. The cycle count repetition setting is only valid when the FlexRay trigger mode is set to FRAME.

**Query Syntax**

```
:SBUS<n>:FLEXray:TRIGger:FRAMe:CCRepetition?
```

The :SBUS<n>:FLEXray:TRIGger:FRAMe:CCRepetition? query returns the current cycle count repetition setting for the FlexRay frame trigger setup.

**Return Format**

```
<cycle_count_repetition><NL>
<cycle_count_repetition> ::= {ALL | <rep #>}
<rep #> ::= integer values 2, 4, 8, 16, 32, or 64
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785

**:SBUS<n>:FLEXray:TRIGger:FRAMe:ID****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:FLEXray:TRIGger:FRAMe:ID <frame_id>`  
`<frame_id> ::= {ALL | <frame #>}`  
`<frame #> ::= integer from 1-2047`

The :SBUS<n>:FLEXray:TRIGger:FRAMe:ID command sets the FlexRay frame ID to trigger on. The frame ID setting is only valid when the FlexRay trigger mode is set to FRAMe.

**Query Syntax**    `:SBUS<n>:FLEXray:TRIGger:FRAMe:ID?`

The :SBUS<n>:FLEXray:TRIGger:FRAMe:ID? query returns the current frame ID setting for the FlexRay frame trigger setup.

**Return Format**    `<frame_id><NL>`  
`<frame_id> ::= {ALL | <frame #>}`  
`<frame #> ::= integer from 1-2047`

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":TRIGger:MODE"](#) on page 1056
- [":SBUS<n>:FLEXray:TRIGger"](#) on page 785

## :SBUS<n>:FLEXray:TRIGger:FRAMe:TYPE

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:FLEXray:TRIGger:FRAMe:TYPE <frame\_type>

```
<frame_type> ::= {NORMal | STARtup | NULL | SYNC | NSTArtup | NNULl |
                  NSYNC | ALL}
```

The :SBUS<n>:FLEXray:TRIGger:FRAMe:TYPE command sets the FlexRay frame type to trigger on. The frame type setting is only valid when the FlexRay trigger mode is set to FRAME.

- NORMal – will trigger on only normal (NSTArtup & NNULl & NSYNC) frames.
- STARtup – will trigger on only startup frames.
- NULL – will trigger on only null frames.
- SYNC – will trigger on only sync frames.
- NSTArtup – will trigger on frames other than startup frames.
- NNULl – will trigger on frames other than null frames.
- NSYNC – will trigger on frames other than sync frames.
- ALL – will trigger on all FlexRay frame types.

**Query Syntax** :SBUS<n>:FLEXray:TRIGger:FRAMe:TYPE?

The :SBUS<n>:FLEXray:TRIGger:FRAMe:TYPE? query returns the current frame type setting for the FlexRay frame trigger setup.

**Return Format** <frame\_type><NL>

```
<frame_type> ::= {NORM | STAR | NULL | SYNC | NSTA | NNUL |
                  NSYN | ALL}
```

**See Also** • "[Introduction to :TRIGger Commands](#)" on page 1047

• "[:TRIGger:MODE](#)" on page 1056

• "[:SBUS<n>:FLEXray:TRIGger](#)" on page 785

## :SBUS<n>:I2S Commands

**NOTE**

These commands are only valid when the I2S serial decode option (Option SND) has been licensed.

**Table 114:**:SBUS<n>:I2S Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:I2S:ALIGnment <setting> (see page 796)	:SBUS<n>:I2S:ALIGnment? (see page 796)	<setting> ::= {I2S   LJ   RJ}
:SBUS<n>:I2S:BASE <base> (see page 797)	:SBUS<n>:I2S:BASE? (see page 797)	<base> ::= {DECimal   HEX}
:SBUS<n>:I2S:CLOCK:SL OPe <slope> (see page 798)	:SBUS<n>:I2S:CLOCK:SL OPe? (see page 798)	<slope> ::= {NEGative   POSitive}
:SBUS<n>:I2S:RWIDTH <receiver> (see page 799)	:SBUS<n>:I2S:RWIDTH? (see page 799)	<receiver> ::= 4-32 in NR1 format
:SBUS<n>:I2S:SOURce:CL OCK <source> (see page 800)	:SBUS<n>:I2S:SOURce:CL OCK? (see page 800)	<source> ::= {CHANnel<n>   EXTERNAL} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:I2S:SOURce:D ATA <source> (see page 801)	:SBUS<n>:I2S:SOURce:D ATA? (see page 801)	<source> ::= {CHANnel<n>   EXTERNAL} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:I2S:SOURce:W SElect <source> (see page 802)	:SBUS<n>:I2S:SOURce:W SElect? (see page 802)	<source> ::= {CHANnel<n>   EXTERNAL} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 114:**:SBUS<n>:I2S Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:I2S:TRIGger <operator> (see <a href="#">page 803</a> )	:SBUS<n>:I2S:TRIGger? (see <a href="#">page 803</a> )	<operator> ::= {EQUAL   NOTequal   LESSthan   GREaterthan   INRange   OUTRange   INCReasing   DECReasing}
:SBUS<n>:I2S:TRIGger: AUDIO <audio_ch> (see <a href="#">page 805</a> )	:SBUS<n>:I2S:TRIGger: AUDio? (see <a href="#">page 805</a> )	<audio_ch> ::= {RIGHT   LEFT   EITHER}
:SBUS<n>:I2S:TRIGger: PATtern:DATA <string> (see <a href="#">page 806</a> )	:SBUS<n>:I2S:TRIGger: PATtern:DATA? (see <a href="#">page 807</a> )	<string> ::= "n" where n ::= 32-bit integer in signed decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   x   \$} when <base> = BINary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   x   \$} when <base> = HEX
:SBUS<n>:I2S:TRIGger: PATtern:FORMAT <base> (see <a href="#">page 808</a> )	:SBUS<n>:I2S:TRIGger: PATtern:FORMAT? (see <a href="#">page 808</a> )	<base> ::= {BINary   HEX   DECimal}
:SBUS<n>:I2S:TRIGger: RANGE <lower>,<upper> (see <a href="#">page 809</a> )	:SBUS<n>:I2S:TRIGger: RANGE? (see <a href="#">page 809</a> )	<lower> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <upper> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal
:SBUS<n>:I2S:TWIDth <word_size> (see <a href="#">page 811</a> )	:SBUS<n>:I2S:TWIDth? (see <a href="#">page 811</a> )	<word_size> ::= 4-32 in NR1 format
:SBUS<n>:I2S:WSLow <low_def> (see <a href="#">page 812</a> )	:SBUS<n>:I2S:WSLow? (see <a href="#">page 812</a> )	<low_def> ::= {LEFT   RIGHT}

## :SBUS<n>:I2S:ALIGnment

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:ALIGnment <setting>`  
`<setting> ::= {I2S | LJ | RJ}`

The :SBUS<n>:I2S:ALIGnment command selects the data alignment of the I2S bus for the serial decoder and/or trigger when in I2S mode:

- I2S – standard.
- LJ – left justified.
- RJ – right justified.

Note that the word select (WS) polarity is specified separately with the :SBUS<n>:I2S:WSLow command.

**Query Syntax**    `:SBUS<n>:I2S:ALIGnment?`

The :SBUS<n>:I2S:ALIGnment? query returns the currently selected I2S data alignment.

**Return Format**    `<setting><NL>`  
`<setting> ::= {I2S | LJ | RJ}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:I2S:CLOCK:SLOPe](#)" on page 798
- "[":SBUS<n>:I2S:RWIDth](#)" on page 799
- "[":SBUS<n>:I2S:TWIDth](#)" on page 811
- "[":SBUS<n>:I2S:WSLow](#)" on page 812

**:SBUS<n>:I2S:BASE**

**N** (see [page 1334](#))

**Command Syntax**    :SBUS<n>:I2S:BASE <base>  
                    <base> ::= {DECimal | HEX}

The :SBUS<n>:I2S:BASE command determines the base to use for the I2S decode display.

**Query Syntax**    :SBUS<n>:I2S:BASE?

The :SBUS<n>:I2S:BASE? query returns the current I2S display decode base.

**Return Format**    <base><NL>  
                    <base> ::= {DECimal | HEX}

**Errors**    - "241, Hardware missing" on page 1293

**See Also**    - "[Introduction to :SBUS<n> Commands](#)" on page 723  
                 - "[:SBUS<n>:I2S Commands](#)" on page 794

**:SBUS<n>:I2S:CLOCK:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:CLOCK:SLOPe <slope>`  
`<slope> ::= {NEGative | POSitive}`

The :SBUS<n>:I2S:CLOCK:SLOPe command specifies which edge of the I2S serial clock signal clocks in data.

- NEGative – Falling edge.
- POSitive – Rising edge.

**Query Syntax**    `:SBUS<n>:I2S:CLOCK:SLOPe?`

The :SBUS<n>:I2S:CLOCK:SLOPe? query returns the current I2S clock slope setting.

**Return Format**    `<slope><NL>`  
`<slope> ::= {NEG | POS}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:I2S:ALIGNment](#)" on page 796
- "[":SBUS<n>:I2S:RWIDth](#)" on page 799
- "[":SBUS<n>:I2S:TWIDth](#)" on page 811
- "[":SBUS<n>:I2S:WSLow](#)" on page 812

**:SBUS<n>:I2S:RWIDth****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:I2S:RWIDth <receiver>`    `<receiver> ::= 4-32 in NR1 format`

The :SBUS<n>:I2S:RWIDth command sets the width of the receiver (decoded) data word in I2S anywhere from 4 bits to 32 bits.

**Query Syntax**    `:SBUS<n>:I2S:RWIDth?`

The :SBUS<n>:I2S:RWIDth? query returns the currently set I2S receiver data word width.

**Return Format**    `<receiver><NL>`    `<receiver> ::= 4-32 in NR1 format`**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047

- "[:SBUS<n>:I2S:ALIGNment](#)" on page 796

- "[:SBUS<n>:I2S:CLOCK:SLOPe](#)" on page 798

- "[:SBUS<n>:I2S:TWIDth](#)" on page 811

- "[:SBUS<n>:I2S:WSLow](#)" on page 812

**:SBUS<n>:I2S:SOURce:CLOCK****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:I2S:SOURce:CLOCK <source>``<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models``<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models``<n> ::= 1 to (# analog channels) in NR1 format``<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :SBUS<n>:I2S:SOURce:CLOCK controls which signal is used as the serial clock (SCLK) source by the serial decoder and/or trigger when in I2S mode.

**Query Syntax**    `:SBUS<n>:I2S:SOURce:CLOCK?`

The :SBUS<n>:I2S:SOURce:CLOCK? query returns the current source for the I2S serial clock (SCLK).

**Return Format**    `<source><NL>`

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":SBUS<n>:I2S:SOURce:DATA](#)" on page 801
  - "[":SBUS<n>:I2S:SOURce:WSELect](#)" on page 802

## :SBUS<n>:I2S:SOURce:DATA

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:I2S:SOURce:DATA <source>

```
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:I2S:SOURce:DATA command controls which signal is used as the serial data (SDATA) source by the serial decoder and/or trigger when in I2S mode.

**Query Syntax** :SBUS<n>:I2S:SOURce:DATA?

The :SBUS<n>:I2S:SOURce:DATA? query returns the current source for the I2S serial data (SDATA).

**Return Format** <source><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:SBUS<n>:I2S:SOURce:CLOCK](#)" on page 800
  - "[:SBUS<n>:I2S:SOURce:WSELect](#)" on page 802

## :SBUS<n>:I2S:SOURce:WSELect

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:I2S:SOURce:WSELect &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   EXTERNAL} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;} for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :SBUS<n>:I2S:SOURce:WSELect command controls which signal is used as the word select (WS) source by the serial decoder and/or trigger when in I2S mode.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:I2S:SOURce:WSELect?</code>
	The :SBUS<n>:I2S:SOURce:WSELect? query returns the current source for I2S word select (WS).
<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• <a href="#">"Introduction to :TRIGger Commands" on page 1047</a></li> <li>• <a href="#">":SBUS&lt;n&gt;:I2S:SOURce:CLOCK" on page 800</a></li> <li>• <a href="#">":SBUS&lt;n&gt;:I2S:SOURce:DATA" on page 801</a></li> </ul>

## :SBUS<n>:I2S:TRIGger

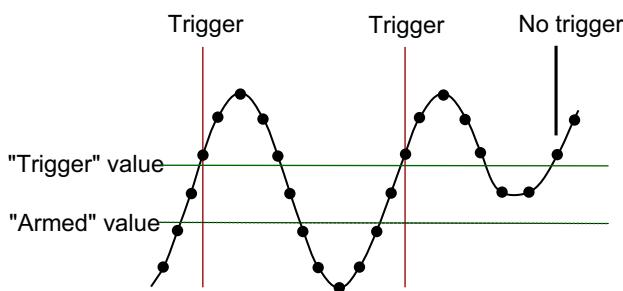
**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:I2S:TRIGger <operator>

```
<operator> ::= {EQUAL | NOTEqual | LESSthan | GREaterthan | INRange
                 | OUTRange | INCReasing | DECReasing}
```

The :SBUS<n>:I2S:TRIGger command sets the I2S trigger operator:

- EQUAL – triggers on the specified audio channel's data word when it equals the specified word.
- NOTEqual – triggers on any word other than the specified word.
- LESSthan – triggers when the channel's data word is less than the specified value.
- GREaterthan – triggers when the channel's data word is greater than the specified value.
- INRange – enter upper and lower values to specify the range in which to trigger.
- OUTRange – enter upper and lower values to specify range in which trigger will not occur.
- INCReasing – triggers when the data value makes a certain increase over time and the specified value is met or exceeded. Use the :SBUS<n>:I2S:TRIGger:RANGE command to set "Trigger" and "Armed" values. The "Trigger" value is the value that must be met or exceeded to cause the trigger. The "Armed" value is the value the data must go below in order to re-arm the oscilloscope (ready it to trigger again).



- DECReasing – similar to INCReasing except the trigger occurs on a certain decrease over time and the "Trigger" data value is less than the "Armed" data value.

**Query Syntax** :SBUS<n>:I2S:TRIGger?

The :SBUS<n>:I2S:TRIGger? query returns the current I2S trigger operator.

<b>Return Format</b>	<operator><NL> <operator> ::= {EQU   NOT   LESS   GRE   INR   OUTR   INCR   DECR}
<b>See Also</b>	<ul style="list-style-type: none"><li>· "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li><li>· "<a href="#">:SBUS&lt;n&gt;:I2S:TRIGger:AUDio</a>" on page 805</li><li>· "<a href="#">:SBUS&lt;n&gt;:I2S:TRIGger:RANGE</a>" on page 809</li><li>· "<a href="#">:SBUS&lt;n&gt;:I2S:TRIGger:PATTern:FORMat</a>" on page 808</li></ul>

**:SBUS<n>:I2S:TRIGger:AUDio**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:TRIGger:AUDio <audio_ch>`  
`<audio_ch> ::= {RIGHT | LEFT | EITHer}`

The :SBUS<n>:I2S:TRIGger:AUDio command specifies the audio channel to trigger on:

- RIGHT – right channel.
- LEFT – left channel.
- EITHer – right or left channel.

**Query Syntax**    `:SBUS<n>:I2S:TRIGger:AUDio?`

The :SBUS<n>:I2S:TRIGger:AUDio? query returns the current audio channel for the I2S trigger.

**Return Format**    `<audio_ch><NL>`  
`<audio_ch> ::= {RIGH | LEFT | EITH}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:I2S:TRIGger](#)" on page 803

## :SBUS<n>:I2S:TRIGger:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:TRIGger:PATTERn:DATA <string>`

```
<string> ::= "n" where n ::= 32-bit integer in signed decimal when
             <base> = DECimal

<string> ::= "nn...n" where n ::= {0 | 1 | x | $} when
             <base> = BINARY

<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $} when
             <base> = HEX
```

### NOTE

<base> is specified with the :SBUS<n>:I2S:TRIGger:PATTERn:FORMat command. The default <base> is DECimal.

The :SBUS<n>:I2S:TRIGger:PATTERn:DATA command specifies the I2S trigger data pattern searched for in each I2S message.

Set a <string> bit to "0" or "1" to set the corresponding bit in the data pattern to low or high, respectively.

Set a <string> bit to "X" to ignore (mask off) that bit in the data pattern.

Use the "\$" character to indicate that the value of the corresponding bit will not be changed (the existing bit value is used).

When <base> = DECimal, the "X" and "\$" characters cannot be entered. When queried, the "\$" character is returned when any bits in the pattern have the value of "X" and <base> = DECimal. When any bits in a given nibble have the value of "X" and <base> = HEX, the "\$" character is returned for the corresponding nibble.

### NOTE

The :SBUS<n>:I2S:TRIGger:PATTERn:DATA command specifies the I2S trigger data pattern used by the EQUal, NOTequal, GREaterthan, and LESSthan trigger conditions. If the GREaterthan or LESSthan trigger condition is selected, the bits specified to be masked off ("X") will be interpreted as 0's.

### NOTE

The length of the trigger data value is determined by the :SBUS<n>:I2S:RWIDth and :SBUS<n>:I2S:TWIDth commands. When the receiver word size is less than the transmitter word size, the data length is equal to the receiver word size. When the receiver word size is greater than the transmitter word size, the data length is equal to the transmitter word size.

### NOTE

If more bits are sent for <string> than the specified trigger data length, the most significant bits will be truncated. If the word size is changed after the <string> is programmed, the added or deleted bits will be added to or deleted from the least significant bits.

**Query Syntax** :SBUS<n>:I2S:TRIGger:PATTERn:DATA?

The :SBUS<n>:I2S:TRIGger:PATTERn:DATA? query returns the currently specified I2S trigger data pattern.

**Return Format** <string><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:SBUS<n>:I2S:TRIGger:PATTERn:FORMat](#)" on page 808
  - "[:SBUS<n>:I2S:TRIGger](#)" on page 803
  - "[:SBUS<n>:I2S:RWIDth](#)" on page 799
  - "[:SBUS<n>:I2S:TWIDth](#)" on page 811
  - "[:SBUS<n>:I2S:TRIGger:AUDio](#)" on page 805

**:SBUS<n>:I2S:TRIGger:PATTERn:FORMAT****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:TRIGger:PATTERn:FORMAT <base>`  
                  `<base> ::= {BINary | HEX | DECimal}`

The :SBUS<n>:I2S:TRIGger:PATTERn:FORMAT command sets the entry (and query) number base used by the :SBUS<n>:I2S:TRIGger:PATTERn:DATA command. The default <base> is DECimal.

**Query Syntax**    `:SBUS<n>:I2S:TRIGger:PATTERn:FORMAT?`

The :SBUS<n>:I2S:TRIGger:PATTERn:FORMAT? query returns the currently set number base for I2S pattern data.

**Return Format**    `<base><NL>`  
                  `<base> ::= {BIN | HEX | DEC}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- "[:SBUS<n>:I2S:TRIGger:AUDio](#)" on page 805
- "[:SBUS<n>:I2S:TRIGger](#)" on page 803

## :SBUS<n>:I2S:TRIGger:RANGE

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:I2S:TRIGger:RANGE <lower>,<upper>
  <lower> ::= 32-bit integer in signed decimal, <nondecimal>
            or <string>
  <upper> ::= 32-bit integer in signed decimal, <nondecimal>,
            or <string>
  <nondecimal> ::= #Hnn...n where n ::= {0,...,9 | A,...,F}
                  for hexadecimal
  <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
  <string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal
```

The :SBUS<n>:I2S:TRIGger:RANGE command sets the lower and upper range boundaries used by the INRange, OUTRange, INCReasing, and DECReasing trigger conditions. You can enter the parameters in any order – the smaller value becomes the <lower> and the larger value becomes the <upper>.

Note that for INCReasing and DECReasing, the <lower> and <upper> values correspond to the "Armed" and "Trigger" softkeys.

### NOTE

The length of the <lower> and <upper> values is determined by the :SBUS<n>:I2S:RWIDth and :SBUS<n>:I2S:TWIDth commands. When the receiver word size is less than the transmitter word size, the length is equal to the receiver word size. When the receiver word size is greater than the transmitter word size, the length is equal to the transmitter word size.

## Query Syntax

:SBUS<n>:I2S:TRIGger:RANGE?

The :SBUS<n>:I2S:TRIGger:RANGE? query returns the currently set lower and upper range boundaries.

## Return Format

```
<lower>,<upper><NL>
  <lower> ::= 32-bit integer in signed decimal, <nondecimal>
            or <string>
  <upper> ::= 32-bit integer in signed decimal, <nondecimal>,
            or <string>
  <nondecimal> ::= #Hnn...n where n ::= {0,...,9 | A,...,F}
                  for hexadecimal
  <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
  <string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal
```

## See Also

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":SBUS<n>:I2S:TRIGger"](#) on page 803
- [":SBUS<n>:I2S:RWIDth"](#) on page 799

- [":SBUS<n>:I2S:TWIDth"](#) on page 811
- [":SBUS<n>:I2S:WSLow"](#) on page 812

**:SBUS<n>:I2S:TWIDth****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:I2S:TWIDth <word_size>`    `<word_size> ::= 4-32 in NR1 format`

The :SBUS<n>:I2S:TWIDth command sets the width of the transmitted data word in I2S anywhere from 4 bits to 32 bits.

**Query Syntax**    `:SBUS<n>:I2S:TWIDth?`

The :SBUS<n>:I2S:TWIDth? query returns the currently set I2S transmitted data word width.

**Return Format**    `<word_size><NL>`    `<word_size> ::= 4-32 in NR1 format`**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047

- [":SBUS<n>:I2S:ALIGNment"](#) on page 796

- [":SBUS<n>:I2S:CLOCK:SLOPe"](#) on page 798

- [":SBUS<n>:I2S:RWIDth"](#) on page 799

- [":SBUS<n>:I2S:WSLow"](#) on page 812

## :SBUS<n>:I2S:WSLow

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:I2S:WSLow <low_def>`  
`<low_def> ::= {LEFT | RIGHT}`

The :SBUS<n>:I2S:WSLow command selects the polarity of the word select (WS) signal:

- LEFT – a word select (WS) state of low indicates left channel data is active on the I2S bus, and a WS state of high indicates right channel data is active on the bus.
- RIGHT – a word select (WS) state of low indicates right channel data is active on the I2S bus, and a WS state of high indicates left channel data is active on the bus.

**Query Syntax**    `:SBUS<n>:I2S:WSLow?`

The :SBUS<n>:I2S:WSLow? query returns the currently selected I2S word select (WS) polarity.

**Return Format**    `<low_def><NL>`  
`<low_def> ::= {LEFT | RIGHT}`

**See Also**    ["Introduction to :TRIGger Commands" on page 1047](#)  
[":SBUS<n>:I2S:ALIGNment" on page 796](#)  
[":SBUS<n>:I2S:CLOCK:SLOPe" on page 798](#)  
[":SBUS<n>:I2S:RWIDth" on page 799](#)  
[":SBUS<n>:I2S:TVIDth" on page 811](#)

## :SBUS<n>:IIC Commands

**NOTE**

These commands are only valid when the low-speed IIC and SPI serial decode option (Option LSS) has been licensed.

**Table 115:**:SBUS<n>:IIC Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:IIC:ASIZE <size> (see page 814)	:SBUS<n>:IIC:ASIZE? (see page 814)	<size> ::= {BIT7   BIT8}
:SBUS<n>:IIC[:SOURce] :CLOCk <source> (see page 815)	:SBUS<n>:IIC[:SOURce] :CLOCk? (see page 815)	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:IIC[:SOURce] :DATA <source> (see page 816)	:SBUS<n>:IIC[:SOURce] :DATA? (see page 816)	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:IIC:TRIGger: PATtern:ADDRess <value> (see page 817)	:SBUS<n>:IIC:TRIGger: PATtern:ADDRess? (see page 817)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger: PATtern:DATA <value> (see page 818)	:SBUS<n>:IIC:TRIGger: PATtern:DATA? (see page 818)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger: PATtern:DATA2 <value> (see page 819)	:SBUS<n>:IIC:TRIGger: PATtern:DATA2? (see page 819)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SBUS<n>:IIC:TRIGger: QUALifier <value> (see page 820)	:SBUS<n>:IIC:TRIGger: QUALifier? (see page 820)	<value> ::= {EQUAL   NOTEQUAL   LESSthan   GREATERthan}
:SBUS<n>:IIC:TRIGger[:TYPE] <type> (see page 821)	:SBUS<n>:IIC:TRIGger[:TYPE]? (see page 821)	<type> ::= {START   STOP   READ7   READEeprom   WRITE7   WRITE10   NACKnowledge   ANACK   R7Data2   W7Data2   REStart}

**:SBUS<n>:IIC:ASIZE****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:IIC:ASIZE <size>`  
                  `<size> ::= {BIT7 | BIT8}`

The :SBUS<n>:IIC:ASIZE command determines whether the Read/Write bit is included as the LSB in the display of the IIC address field of the decode bus.

**Query Syntax**    `:SBUS<n>:IIC:ASIZE?`

The :SBUS<n>:IIC:ASIZE? query returns the current IIC address width setting.

**Return Format**    `<mode><NL>`  
                  `<mode> ::= {BIT7 | BIT8}`

**Errors**    - ["-241, Hardware missing" on page 1293](#)

**See Also**    - ["Introduction to :SBUS<n> Commands" on page 723](#)  
                  - [":SBUS<n>:IIC Commands" on page 813](#)

**:SBUS<n>:IIC[:SOURce]:CLOCK**

**N** (see [page 1334](#))

- Command Syntax**    `:SBUS<n>:IIC: [SOURce:] CLOCk <source>`
- `<source> ::= {CHANnel<n> | EXTernal} for the DSO models`
- `<source> ::= {CHANnel<n> | DIGital<d>} for the MSO models`
- `<n> ::= 1 to (# analog channels) in NR1 format`
- `<d> ::= 0 to (# digital channels - 1) in NR1 format`
- The :SBUS<n>:IIC:[SOURce:]CLOCK command sets the source for the IIC serial clock (SCL).
- Query Syntax**    `:SBUS<n>:IIC: [SOURce:] CLOCk?`
- The :SBUS<n>:IIC:[SOURce:]CLOCK? query returns the current source for the IIC serial clock.
- Return Format**    `<source><NL>`
- See Also**
  - "Introduction to :TRIGger Commands" on page 1047
  - "[:SBUS<n>:IIC\[:SOURce\]:DATA](#)" on page 816

**:SBUS<n>:IIC[:SOURce]:DATA****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:IIC[:SOURce:] DATA <source>`

`<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models`

`<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models`

`<n> ::= 1 to (# analog channels) in NR1 format`

`<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :SBUS<n>:IIC[:SOURce:]DATA command sets the source for IIC serial data (SDA).

**Query Syntax**    `:SBUS<n>:IIC[:SOURce:] DATA?`

The :SBUS<n>:IIC[:SOURce:]DATA? query returns the current source for IIC serial data.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:IIC\[:SOURce\]:CLOCK](#)" on page 815

## :SBUS<n>:IIC:TRIGger:PATTERn:ADDResS

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:IIC:TRIGger:PATTERn:ADDResS <value>

<value> ::= integer or <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}

The :SBUS<n>:IIC:TRIGger:PATTERn:ADDResS command sets the address for IIC data. The address can range from 0x00 to 0x7F (7-bit) or 0x3FF (10-bit) hexadecimal. Use the don't care address (-1 or 0xFFFFFFFF) to ignore the address value.

**Query Syntax** :SBUS<n>:IIC:TRIGger:PATTERn:ADDResS?

The :SBUS<n>:IIC:TRIGger:PATTERn:ADDResS? query returns the current address for IIC data.

**Return Format** <value><NL>

<value> ::= integer

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:IIC:TRIGger:PATTERn:DATA](#)" on page 818
- "[":SBUS<n>:IIC:TRIGger:PATTERn:DATa2](#)" on page 819
- "[":SBUS<n>:IIC:TRIGger\[:TYPE\]](#)" on page 821

**:SBUS<n>:IIC:TRIGger:PATTERn:DATA**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:IIC:TRIGger:PATTERn:DATA <value>`

`<value> ::= integer or <string>`

`<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}`

The :SBUS<n>:IIC:TRIGger:PATTERn:DATA command sets IIC data. The data value can range from 0x00 to 0xFF (hexadecimal). Use the don't care data pattern (-1 or 0xFFFFFFFF) to ignore the data value.

**Query Syntax**    `:SBUS<n>:IIC:TRIGger:PATTERn:DATA?`

The :SBUS<n>:IIC:TRIGger:PATTERn:DATA? query returns the current pattern for IIC data.

**Return Format**    `<value><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:IIC:TRIGger:PATTERn:ADDResS](#)" on page 817
- "[:SBUS<n>:IIC:TRIGger:PATTERn:DATa2](#)" on page 819
- "[:SBUS<n>:IIC:TRIGger\[:TYPE\]](#)" on page 821

## :SBUS<n>:IIC:TRIGger:PATTERn:DATA2

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:IIC:TRIGger:PATTERn:DATA2 <value>

<value> ::= integer or <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}

The :SBUS<n>:IIC:TRIGger:PATTERn:DATA2 command sets IIC data 2. The data value can range from 0x00 to 0x0FF (hexadecimal). Use the don't care data pattern (-1 or 0xFFFFFFFF) to ignore the data value.

**Query Syntax** :SBUS<n>:IIC:TRIGger:PATTERn:DATA2?

The :SBUS<n>:IIC:TRIGger:PATTERn:DATA2? query returns the current pattern for IIC data 2.

**Return Format** <value><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:SBUS<n>:IIC:TRIGger:PATTERn:ADDResS](#)" on page 817
  - "[:SBUS<n>:IIC:TRIGger:PATTERn:DATA](#)" on page 818
  - "[:SBUS<n>:IIC:TRIGger\[:TYPE\]](#)" on page 821

**:SBUS<n>:IIC:TRIGger:QUALifier**

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:IIC:TRIGger:QUALifier &lt;value&gt;</code> <code>&lt;value&gt; ::= {EQUAL   NOTEqual   LESSthan   GREaterthan}</code>
	The :SBUS<n>:IIC:TRIGger:QUALifier command sets the IIC data qualifier when TRIGger:IIC:TRIGger[:TYPE] is set to READEprom.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:IIC:TRIGger:QUALifier?</code>
	The :SBUS<n>:IIC:TRIGger:QUALifier? query returns the current IIC data qualifier value.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code> <code>&lt;value&gt; ::= {EQUAL   NOTEqual   LESSthan   GREaterthan}</code>
<b>See Also</b>	<ul style="list-style-type: none"><li><a href="#">"Introduction to :TRIGger Commands"</a> on page 1047</li><li><a href="#">":TRIGger:MODE"</a> on page 1056</li><li><a href="#">":SBUS&lt;n&gt;:IIC:TRIGger[:TYPE]"</a> on page 821</li></ul>

## :SBUS<n>:IIC:TRIGger[:TYPE]

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:IIC:TRIGger[:TYPE] <value>

```
<value> ::= {START | STOP | READ7 | READEprom | WRITe7 | WRITe10
| NACKnowledge | ANACK | R7Data2 | W7Data2 | RESTart}
```

The :SBUS<n>:IIC:TRIGger[:TYPE] command sets the IIC trigger type:

- START – Start condition.
- STOP – Stop condition.
- READ7 – 7-bit address frame containing (Start:Address7:Read:Ack:Data). The value READ is also accepted for READ7.
- R7Data2 – 7-bit address frame containing (Start:Address7:Read:Ack:Data:Ack:Data2).
- READEprom – EEPROM data read.
- WRITe7 – 7-bit address frame containing (Start:Address7:Write:Ack:Data). The value WRITe is also accepted for WRITe7.
- W7Data2 – 7-bit address frame containing (Start:Address7:Write:Ack:Data:Ack:Data2).
- WRITe10 – 10-bit address frame containing (Start:Address byte1:Write:Ack:Address byte 2:Data).
- NACKnowledge – Missing acknowledge.
- ANACK – Address with no acknowledge.
- RESTart – Another start condition occurs before a stop condition.

### NOTE

The short form of READ7 (READ7), READEprom (READE), WRITe7 (WRIT7), and WRITe10 (WRIT10) do not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)).

**Query Syntax** :SBUS<n>:IIC:TRIGger[:TYPE] ?

The :SBUS<n>:IIC:TRIGger[:TYPE]? query returns the current IIC trigger type value.

**Return Format** <value><NL>

```
<value> ::= {STAR | STOP | READ7 | READE | WRIT7 | WRIT10 | NACK | ANAC
| R7D2 | W7D2 | REST}
```

**See Also** • "[Introduction to :TRIGger Commands](#)" on page 1047

• "[:TRIGger:MODE](#)" on page 1056

• "[:SBUS<n>:IIC:TRIGger:PATTERn:ADDReSS](#)" on page 817

• "[:SBUS<n>:IIC:TRIGger:PATTERn:DATA](#)" on page 818

- "[:SBUS<n>:IIC:TRIGger:PATTERn:DATa2](#)" on page 819
- "[:SBUS<n>:IIC:TRIGger:QUALifier](#)" on page 820
- "["Long Form to Short Form Truncation Rules"](#) on page 1336

## :SBUS<n>:LIN Commands

**NOTE**

These commands are valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

**Table 116**:SBUS<n>:LIN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:LIN:DISPLAY <type> (see <a href="#">page 825</a> )	:SBUS<n>:LIN:DISPLAY? (see <a href="#">page 825</a> )	<type> ::= {HEXAdecimal   SYMBolic}
:SBUS<n>:LIN:PARity { {0   OFF}   {1   ON} } (see <a href="#">page 826</a> )	:SBUS<n>:LIN:PARity? (see <a href="#">page 826</a> )	{0   1}
:SBUS<n>:LIN:SAMPLEpo int <value> (see <a href="#">page 827</a> )	:SBUS<n>:LIN:SAMPLEpo int? (see <a href="#">page 827</a> )	<value> ::= {60   62.5   68   70   75   80   87.5} in NR3 format
:SBUS<n>:LIN:SIGNAL:B AUDrate <baudrate> (see <a href="#">page 828</a> )	:SBUS<n>:LIN:SIGNAL:B AUDrate? (see <a href="#">page 828</a> )	<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments
:SBUS<n>:LIN:SOURce <source> (see <a href="#">page 829</a> )	:SBUS<n>:LIN:SOURce? (see <a href="#">page 829</a> )	<source> ::= {CHANnel<n>   EXTernal} for DSO models <source> ::= {CHANnel<n>   DIGital<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:LIN:STANDARD <std> (see <a href="#">page 830</a> )	:SBUS<n>:LIN:STANDARD ? (see <a href="#">page 830</a> )	<std> ::= {LIN13   LIN20}
:SBUS<n>:LIN:SYNCbre ak <value> (see <a href="#">page 831</a> )	:SBUS<n>:LIN:SYNCbre ak? (see <a href="#">page 831</a> )	<value> ::= integer = {11   12   13}
:SBUS<n>:LIN:TRIGger <condition> (see <a href="#">page 832</a> )	:SBUS<n>:LIN:TRIGger? (see <a href="#">page 832</a> )	<condition> ::= {SYNCbreak   ID   DATA}

**Table 116:**:SBUS< n >:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS< n >:LIN:TRIGger: ID <value> (see <a href="#">page 833</a> )	:SBUS< n >:LIN:TRIGger: ID? (see <a href="#">page 833</a> )	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9   A,...,F} for hexadecimal
:SBUS< n >:LIN:TRIGger: PATtern:DATA <string> (see <a href="#">page 834</a> )	:SBUS< n >:LIN:TRIGger: PATtern:DATA? (see <a href="#">page 834</a> )	<string> ::= "n" where n ::= 32-bit integer in unsigned decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   X   \$} when <base> = BINary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$} when <base> = HEX
:SBUS< n >:LIN:TRIGGER: PATtern:DATA:LENGTH <length> (see <a href="#">page 836</a> )	:SBUS< n >:LIN:TRIGger: PATtern:DATA:LENGTH? (see <a href="#">page 836</a> )	<length> ::= integer from 1 to 8 in NR1 format
:SBUS< n >:LIN:TRIGger: PATtern:FORMAT <base> (see <a href="#">page 837</a> )	:SBUS< n >:LIN:TRIGger: PATtern:FORMAT? (see <a href="#">page 837</a> )	<base> ::= {BINary   HEX   DECimal}
:SBUS< n >:LIN:TRIGger: SYMBolic:FRAMe <name> (see <a href="#">page 838</a> )	:SBUS< n >:LIN:TRIGger: SYMBolic:FRAMe? (see <a href="#">page 838</a> )	<name> ::= quoted ASCII string
:SBUS< n >:LIN:TRIGger: SYMBolic:SIGNal <name> (see <a href="#">page 839</a> )	:SBUS< n >:LIN:TRIGger: SYMBolic:SIGNal? (see <a href="#">page 839</a> )	<name> ::= quoted ASCII string
:SBUS< n >:LIN:TRIGger: SYMBolic:VALue <data> (see <a href="#">page 840</a> )	:SBUS< n >:LIN:TRIGger: SYMBolic:VALue? (see <a href="#">page 840</a> )	<data> ::= value in NR3 format

**:SBUS<n>:LIN:DISPlay****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:DISPlay <type>`  
                  `<type> ::= {HEXAdecimal | SYMBolic}`

The :SBUS<n>:LIN:DISPlay command specifies, when LIN symbolic data is loaded into the oscilloscope, whether symbolic values (from the LDF file) or hexadecimal values are displayed in the decode waveform and the Lister window.

**Query Syntax**    `:SBUS<n>:LIN:DISPlay?`

The :SBUS<n>:LIN:DISPlay? query returns the LIN decode display type.

**Return Format**    `<type><NL>`  
                  `<type> ::= {HEX | SYMB}`

**See Also**    • " [":RECall:LDF\[:STARt\]](#)" on page 688

**:SBUS<n>:LIN:PARity****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:PARity <display>`  
                  `<display> ::= {{1 | ON} | {0 | OFF}}`

The :SBUS<n>:LIN:PARity command determines whether the parity bits are included as the most significant bits (MSB) in the display of the Frame Id field in the LIN decode bus.

**Query Syntax**    `:SBUS<n>:LIN:PARity?`

The :SBUS<n>:LIN:PARity? query returns the current LIN parity bits display setting of the serial decode bus.

**Return Format**    `<display><NL>`  
                  `<display> ::= {0 | 1}`

**Errors**    • ["-241, Hardware missing"](#) on page 1293

**See Also**    • ["Introduction to :SBUS<n> Commands"](#) on page 723  
                  • [":SBUS<n>:LIN Commands"](#) on page 823

## :SBUS<n>:LIN:SAMPLEpoint

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:SAMPLEpoint <value>`

`<value><NL>`

`<value> ::= { 60 | 62.5 | 68 | 70 | 75 | 80 | 87.5 } in NR3 format`

The :SBUS<n>:LIN:SAMPLEpoint command sets the point during the bit time where the bit level is sampled to determine whether the bit is dominant or recessive. The sample point represents the percentage of time between the beginning of the bit time to the end of the bit time.

### NOTE

The sample point values are not limited by the baud rate.

**Query Syntax**    `:SBUS<n>:LIN:SAMPLEpoint?`

The :SBUS<n>:LIN:SAMPLEpoint? query returns the current LIN sample point setting.

**Return Format**    `<value><NL>`

`<value> ::= { 60 | 62.5 | 68 | 70 | 75 | 80 | 87.5 } in NR3 format`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832

**:SBUS<n>:LIN:SIGNAl:BAUDrate****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:SIGNAl:BAUDrate <baudrate>`  
`<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments`

The :SBUS<n>:LIN:SIGNAl:BAUDrate command sets the standard baud rate of the LIN signal from 2400 b/s to 625 kb/s in 100 b/s increments. If you enter a baud rate that is not divisible by 100 b/s, the baud rate is set to the nearest baud rate divisible by 100 b/s.

**Query Syntax**    `:SBUS<n>:LIN:SIGNAl:BAUDrate?`

The :SBUS<n>:LIN:SIGNAl:BAUDrate? query returns the current LIN baud rate setting.

**Return Format**    `<baudrate><NL>`  
`<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:SIGNAl:DEFinition](#)" on page 1284
- "[:SBUS<n>:LIN:SOURce](#)" on page 829

## :SBUS<n>:LIN:SOURce

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:SOURce <source>`

```
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:LIN:SOURce command sets the source for the LIN signal.

**Query Syntax**    `:SBUS<n>:LIN:SOURce?`

The :SBUS<n>:LIN:SOURce? query returns the current source for the LIN signal.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:SIGNAl:DEFinition](#)" on page 1284

**:SBUS<n>:LIN:STANDARD****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:STANDARD <std>`  
                  `<std> ::= {LIN13 | LIN20}`

The :SBUS<n>:LIN:STANDARD command sets the LIN standard in effect for triggering and decoding to be LIN1.3 or LIN2.0.

**Query Syntax**    `:SBUS<n>:LIN:STANDARD?`

The :SBUS<n>:LIN:STANDARD? query returns the current LIN standard setting.

**Return Format**    `<std><NL>`  
                  `<std> ::= {LIN13 | LIN20}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:LIN:SIGNAl:DEFinition](#)" on page 1284
- "[:SBUS<n>:LIN:SOURce](#)" on page 829

## :SBUS<n>:LIN:SYNCbreak

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:SYNCbreak <value>`

`<value> ::= integer = {11 | 12 | 13}`

The :SBUS<n>:LIN:SYNCbreak command sets the length of the LIN sync break to be greater than or equal to 11, 12, or 13 clock lengths. The sync break is the idle period in the bus activity at the beginning of each packet that distinguishes one information packet from the previous one.

**Query Syntax**    `:SBUS<n>:LIN:SYNCbreak?`

The :SBUS<n>:LIN:SYNCbreak? query returns the current LIN sync break setting.

**Return Format**    `<value><NL>`

`<value> ::= {11 | 12 | 13}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:MODE](#)" on page 1056
- "[":SBUS<n>:LIN:SIGNAl:DEFinition](#)" on page 1284
- "[":SBUS<n>:LIN:SOURce](#)" on page 829

## :SBUS<n>:LIN:TRIGger

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:TRIGger <condition>`

`<condition> ::= {SYNCbreak | ID | DATA | PARityerror | CSUMerror  
| FRAMe | FSIGnal}`

The :SBUS<n>:LIN:TRIGger command sets the LIN trigger condition to be:

- SYNCbreak – Sync Break.
  - ID – Frame ID.
- Use the :SBUS<n>:LIN:TRIGger:ID command to specify the frame ID.
- DATA – Frame ID and Data.
- Use the :SBUS<n>:LIN:TRIGger:ID command to specify the frame ID.

Use the :SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGTH and :SBUS<n>:LIN:TRIGger:PATTERn:DATA commands to specify the data string length and value.

- PARityerror – parity errors.
- CSUMerror – checksum errors.
- FRAMe – Triggers on a symbolic frame.
- FSIGnal – Triggers on a symbolic frame and a signal value.

**Query Syntax**    `:SBUS<n>:LIN:TRIGger?`

The :SBUS<n>:LIN:TRIGger? query returns the current LIN trigger value.

**Return Format**    `<condition><NL>`

`<condition> ::= {SYNC | ID | DATA | PAR | CSUM | FRAM | FSIG}`

- Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :TRIGger Commands" on page 1047](#)  
 • [":TRIGger:MODE" on page 1056](#)  
 • [":SBUS<n>:LIN:TRIGger:ID" on page 833](#)  
 • [":SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGTH" on page 836](#)  
 • [":SBUS<n>:LIN:TRIGger:PATTERn:DATA" on page 834](#)  
 • [":SBUS<n>:LIN:SIGNal:DEFinition" on page 1284](#)  
 • [":SBUS<n>:LIN:SOURce" on page 829](#)  
 • [":RECall:LDF\[:START\]" on page 688](#)  
 • [":SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe" on page 838](#)  
 • [":SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal" on page 839](#)  
 • [":SBUS<n>:LIN:TRIGger:SYMBolic:VALUe" on page 840](#)

## :SBUS<n>:LIN:TRIGger:ID

**N** (see [page 1334](#))

**Command Syntax**    :SBUS<n>:LIN:TRIGger:ID <value>

```
<value> ::= 7-bit integer in decimal, <nondecimal>, or <string>
           from 0-63 or 0x00-0x3f

<nondecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal
<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F} for hexadecimal
```

The :SBUS<n>:LIN:TRIGger:ID command defines the LIN identifier searched for in each CAN message when the LIN trigger mode is set to frame ID.

Setting the ID to a value of "-1" results in "0xXX" which is equivalent to all IDs.

**Query Syntax**    :SBUS<n>:LIN:TRIGger:ID?

The :SBUS<n>:LIN:TRIGger:ID? query returns the current LIN identifier setting.

**Return Format**    <value><NL>

<value> ::= integer in decimal

**Errors**    • ["-241, Hardware missing"](#) on page 1293

**See Also**    • ["Introduction to :TRIGger Commands"](#) on page 1047  
               • [":TRIGger:MODE"](#) on page 1056  
               • [":SBUS<n>:LIN:TRIGger"](#) on page 832  
               • [":SBUS<n>:LIN:SIGNAl:DEFinition"](#) on page 1284  
               • [":SBUS<n>:LIN:SOURce"](#) on page 829

## :SBUS<n>:LIN:TRIGger:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:LIN:TRIGger:PATTERn:DATA <string>
<string> ::= "n" where n ::= 32-bit integer in unsigned decimal when
            <base> = DECimal
<string> ::= "nn...n" where n ::= {0 | 1 | x | $} when
            <base> = BINary
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $} when
            <base> = HEX
```

**NOTE**

<base> is specified with the :SBUS<n>:LIN:TRIGger:PATTERn:FORMat command. The default <base> is BINary.

The :SBUS<n>:LIN:TRIGger:PATTERn:DATA command specifies the LIN trigger data pattern searched for in each LIN data field.

Set a <string> bit to "0" or "1" to set the corresponding bit in the data pattern to low or high, respectively.

Set a <string> bit to "X" to ignore (mask off) that bit in the data pattern.

Use the "\$" character to indicate that the value of the corresponding bit will not be changed (the existing bit value is used).

When <base> = DECimal, the "X" and "\$" characters cannot be entered. When queried, the "\$" character is returned when any bits in the pattern have the value of "X" and <base> = DECimal. When any bits in a given nibble have the value of "X" and <base> = HEX, the "\$" character is returned for the corresponding nibble.

**NOTE**

The length of the trigger data value is determined by the :SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGTH command.

**NOTE**

If more bits are sent for <string> than the specified trigger pattern data length, the most significant bits will be truncated. If the data length size is changed after the <string> is programmed, the added or deleted bits will be added to or deleted from the least significant bits.

**Query Syntax**

```
:SBUS<n>:LIN:TRIGger:PATTERn:DATA?
```

The :SBUS<n>:LIN:TRIGger:PATTERn:DATA? query returns the currently specified LIN trigger data pattern.

**Return Format**

```
<string><NL>
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:LIN:TRIGger:PTT:FORMAT](#)" on page 837
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:TRIGger:PTT:DATA:LENGTH](#)" on page 836

**:SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth <length>`    `<length> ::= integer from 1 to 8 in NR1 format`

The :SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth command sets the number of 8-bit bytes in the LIN data string. The number of bytes in the string can be anywhere from 1 bytes to 8 bytes (64 bits). The value for these bytes is set by the :SBUS<n>:LIN:TRIGger:PATTERn:DATA command.

**Query Syntax**    `:SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth?`

The :SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth? query returns the current LIN data pattern length setting.

**Return Format**    `<count><NL>`    `<count> ::= integer from 1 to 8 in NR1 format`**Errors**    • ["-241, Hardware missing"](#) on page 1293**See Also**    • ["Introduction to :TRIGger Commands"](#) on page 1047    • [":SBUS<n>:LIN:TRIGger:PATTERn:DATA"](#) on page 834    • [":SBUS<n>:LIN:SOURce"](#) on page 829

## :SBUS<n>:LIN:TRIGger:PATTERn:FORMAT

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:TRIGger:PATTERn:FORMAT <base>`  
`<base> ::= {BINary | HEX | DECimal}`

The :SBUS<n>:LIN:TRIGger:PATTERn:FORMAT command sets the entry (and query) number base used by the :SBUS<n>:LIN:TRIGger:PATTERn:DATA command. The default <base> is BINary.

**Query Syntax**    `:SBUS<n>:LIN:TRIGger:PATTERn:FORMAT?`

The :SBUS<n>:LIN:TRIGger:PATTERn:FORMAT? query returns the currently set number base for LIN pattern data.

**Return Format**    `<base><NL>`  
`<base> ::= {BIN | HEX | DEC}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:LIN:TRIGger:PATTERn:DATA](#)" on page 834
- "[:SBUS<n>:LIN:TRIGger:PATTERn:DATA:LENGth](#)" on page 836

**:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe <name>`  
                  `<name> ::= quoted ASCII string`

The :SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe command specifies the message to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FRAMe or FSIGnal.

**Query Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe?`

The :SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe? query returns the specified message.

**Return Format**    `<name><NL>`  
                  `<name> ::= quoted ASCII string`

**See Also**

- "[:RECall:LDF\[:STARt\]](#)" on page 688
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal](#)" on page 839
- "[:SBUS<n>:LIN:TRIGger:SYMBolic:VALue](#)" on page 840

## :SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal <name>`  
`<name> ::= quoted ASCII string`

The :SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal command specifies the signal to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FSIGnal.

**Query Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal?`

The :SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal? query returns the specified signal.

**Return Format**    `<name><NL>`  
`<name> ::= quoted ASCII string`

**See Also**

- "[:RECall:LDF\[:STARt\]](#)" on page 688
- "[:SBUS<n>:LIN:TRIGger](#)" on page 832
- "[:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe](#)" on page 838
- "[:SBUS<n>:LIN:TRIGger:SYMBolic:VALue](#)" on page 840

## :SBUS<n>:LIN:TRIGger:SYMBolic:VALue

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:VALue <data>`  
`<data> ::= value in NR3 format`

The :SBUS<n>:LIN:TRIGger:SYMBolic:VALue command specifies the signal value to trigger on when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN trigger mode is set to FSIGnal.

### NOTE

Encoded signal values are not supported in the remote interface (even though they can be used in the front panel graphical interface).

**Query Syntax**    `:SBUS<n>:LIN:TRIGger:SYMBolic:VALue?`

The :SBUS<n>:LIN:TRIGger:SYMBolic:VALue? query returns the specified signal value.

**Return Format**    `<data><NL>`

`<data> ::= value in NR3 format`

**See Also**

- "[:RECall:LDF\[:START\]](#)" on page 688
- "[":SBUS<n>:LIN:TRIGger](#)" on page 832
- "[":SBUS<n>:LIN:TRIGger:SYMBolic:FRAME](#)" on page 838
- "[":SBUS<n>:LIN:TRIGger:SYMBolic:SIGNAl](#)" on page 839

## :SBUS<n>:M1553 Commands

**NOTE**

These commands are valid when the DSOX4AERO MIL-STD-1553 and ARINC 429 triggering and serial decode option (Option AERO) has been licensed.

**Table 117** :SBUS<n>:M1553 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:M1553:AUTose tup (see <a href="#">page 842</a> )	n/a	n/a
:SBUS<n>:M1553:BASE <base> (see <a href="#">page 843</a> )	:SBUS<n>:M1553:BASE? (see <a href="#">page 843</a> )	<base> ::= {BINary   HEX}
:SBUS<n>:M1553:SOURce <source> (see <a href="#">page 844</a> )	:SBUS<n>:M1553:SOURce? (see <a href="#">page 844</a> )	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:M1553:TRIGge r:PATTERn:DATA <string> (see <a href="#">page 845</a> )	:SBUS<n>:M1553:TRIGge r:PATTERn:DATA? (see <a href="#">page 845</a> )	<string> ::= "nn...n" where n ::= {0   1   X}
:SBUS<n>:M1553:TRIGge r:RTA <value> (see <a href="#">page 846</a> )	:SBUS<n>:M1553:TRIGge r:RTA? (see <a href="#">page 846</a> )	<value> ::= 5-bit integer in decimal, <nondecimal>, or <string> from 0-31 <nondecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F}
:SBUS<n>:M1553:TRIGge r:TYPE <type> (see <a href="#">page 847</a> )	:SBUS<n>:M1553:TRIGge r:TYPE? (see <a href="#">page 847</a> )	<type> ::= {DSTArt   DSTOp   CSTArt   CSTOp   RTA   PERRor   SERRor   MERRor   RTA11}

**:SBUS<n>:M1553:AUTosetup**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:M1553:TRIGger:AUTosetup`

The `:SBUS<n>:M1553:AUTosetup` command automatically sets these options for decoding and triggering on MIL-STD-1553 signals:

- High/Low Trigger Thresholds: to a voltage value equal to  $\pm 1/3$  division based on the source channel's current V/div setting.
- Noise Reject: Off.
- Probe Attenuation: 10.0.
- Serial Decode: On.
- Trigger: the specified serial bus (n of `SBUS<n>`).

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:M1553:SOURce](#)" on page 844

**:SBUS<n>:M1553:BASE****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:M1553:BASE <base>`  
                  `<base> ::= {BINary | HEX}`

The :SBUS<n>:M1553:BASE command determines the base to use for the MIL-STD-1553 decode display.

**Query Syntax**    `:SBUS<n>:M1553:BASE?`

The :SBUS<n>:M1553:BASE? query returns the current MIL-STD-1553 display decode base.

**Return Format**    `<base><NL>`  
                  `<base> ::= {BIN | HEX}`

**Errors**    • "241, Hardware missing" on page 1293

**See Also**    • "Introduction to :SBUS<n> Commands" on page 723  
                  • ":SBUS<n>:M1553 Commands" on page 841

**:SBUS<n>:M1553:SOURce****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:M1553:SOURce <source>`    `<source> ::= {CHANnel<n>}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :SBUS<n>:M1553:SOURce command sets the source of the MIL-STD 1553 signal.

Use the :TRIGger:LEVel:HIGH and :TRIGger:LEVel:LOW commands to set the threshold levels for the selected source.

**Query Syntax**    `:SBUS<n>:M1553:TRIGger:SOURce?`

The :SBUS<n>:M1553:SOURce? query returns the currently set source of the MIL-STD 1553 signal.

**Return Format**    `<source><NL>`    `<source> ::= {CHAN<n>}`    `<n> ::= 1 to (# analog channels) in NR1 format`**See Also**

- "[:TRIGger:LEVel:HIGH](#)" on page 1054

- "[:TRIGger:LEVel:LOW](#)" on page 1055

- "[:TRIGger:MODE](#)" on page 1056

- "[Introduction to :TRIGger Commands](#)" on page 1047

## :SBUS<n>:M1553:TRIGger:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:M1553:TRIGger:PATTERn:DATA <string>`  
`<string> ::= "nn...n" where n ::= {0 | 1 | x}`

The :SBUS<n>:M1553:TRIGger:PATTERn:DATA command sets the 11 bits to trigger on if the trigger type has been set to RTA11 (RTA + 11 Bits) using the :SBUS<n>:M1553:TRIGger:TYPE command.

**Query Syntax**    `:SBUS<n>:M1553:TRIGger:PATTERn:DATA?`

The :SBUS<n>:M1553:TRIGger:PATTERn:DATA? query returns the current 11-bit setting.

**Return Format**    `<string><NL>`  
`<string> ::= "nn...n" where n ::= {0 | 1 | x}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:M1553:TRIGger:TYPE](#)" on page 847
- "[":SBUS<n>:M1553:TRIGger:RTA](#)" on page 846

**:SBUS<n>:M1553:TRIGger:RTA****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:M1553:TRIGger:RTA <value>`

```
<value> ::= 5-bit integer in decimal, <nondecimal>, or  
<string> from 0-31  
  
<nondecimal> ::= #Hnn where n ::= {0,...,9|A,...,F}  
  
<string> ::= "0xnn" where n ::= {0,...,9|A,...,F}
```

The :SBUS<n>:M1553:TRIGger:RTA command sets the Remote Terminal Address (RTA) to trigger on when the trigger type has been set to RTA or RTA11 (using the :SBUS<n>:M1553:TRIGger:TYPE command).

To set the RTA value to don't cares (0xXX), set the value to -1.

**Query Syntax**    `:SBUS<n>:M1553:TRIGger:RTA?`

The :SBUS<n>:M1553:TRIGger:RTA? query returns the RTA value.

**Return Format**    `<value><NL>` in decimal format**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:M1553:TRIGger:TYPE](#)" on page 847

## :SBUS<n>:M1553:TRIGger:TYPE

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:M1553:TRIGger:TYPE <type>

```
<type> ::= {DSTArt | DSTOp | CSTArt | CSTOp | RTA | PERRor | SERRor
            | MERRor | RTA11}
```

The :SBUS<n>:M1553:TRIGger:TYPE command specifies the type of MIL-STD-1553 trigger to be used:

- DSTArt – (Data Word Start) triggers on the start of a Data word (at the end of a valid Data Sync pulse).
- DSTOp – (Data Word Stop) triggers on the end of a Data word.
- CSTArt – (Command/Status Word Start) triggers on the start of Command/Status word (at the end of a valid C/S Sync pulse).
- CSTOp – (Command/Status Word Stop) triggers on the end of a Command/Status word.
- RTA – (Remote Terminal Address) triggers if the RTA of the Command/Status word matches the specified value. The value is specified in hex.
- RTA11 – (RTA + 11 Bits) triggers if the RTA and the remaining 11 bits match the specified criteria. The RTA can be specified as a hex value, and the remaining 11 bits can be specified as a 1, 0, or X (don't care).
- PERRor – (Parity Error) triggers if the (odd) parity bit is incorrect for the data in the word.
- MERRor – (Manchester Error) triggers if a Manchester encoding error is detected.
- SERRor – (Sync Error) triggers if an invalid Sync pulse is found.

**Query Syntax** :SBUS<n>:M1553:TRIGger:TYPE?

The :SBUS<n>:M1553:TRIGger:TYPE? query returns the currently set MIL-STD-1553 trigger type.

**Return Format** <type><NL>

```
<type> ::= {DSTA | DSTO | CSTA | CSTO | RTA | PERR | SERR
            | MERR | RTA11}
```

**See Also** • ["Introduction to :TRIGger Commands"](#) on page 1047

• [":SBUS<n>:M1553:TRIGger:RTA"](#) on page 846

• [":SBUS<n>:M1553:TRIGger:PATTERn:DATA"](#) on page 845

• [":TRIGger:MODE"](#) on page 1056

## :SBUS< n >:SENT Commands

**NOTE**

These commands are valid when the automotive SENT serial decode and triggering option has been licensed.

**Table 118:**:SBUS< n >:SENT Commands Summary

Command	Query	Options and Query Returns
:SBUS< n >:SENT:CLOCK <period> (see page 851)	:SBUS< n >:SENT:CLOCK? (see page 851)	<period> ::= the nominal clock period (tick), from 1 us to 300 us, in NR3 format.
:SBUS< n >:SENT:CRC <format> (see page 852)	:SBUS< n >:SENT:CRC? (see page 852)	<format> ::= {LEGacy   RECommended}
:SBUS< n >:SENT:DISPlay <base> (see page 853)	:SBUS< n >:SENT:DISPlay? (see page 853)	<base> ::= {HEX   DECimal   SYMBolic}
:SBUS< n >:SENT:FORMAT <decode> (see page 855)	:SBUS< n >:SENT:FORMAT? (see page 855)	<decode> ::= {NIBBles   FSIGnal   FSSerial   FESerial   SSERial   ESErial}
:SBUS< n >:SENT:IDLE <state> (see page 857)	:SBUS< n >:SENT:IDLE? (see page 857)	<state> ::= {LOW   HIGH}
:SBUS< n >:SENT:LENGTH <#_nibbles> (see page 858)	:SBUS< n >:SENT:LENGTH? (see page 858)	<#_nibbles> ::= from 1-6, in NR1 format.
:SBUS< n >:SENT:PPULse { {0   OFF}   {1   ON} } (see page 859)	:SBUS< n >:SENT:PPULse? (see page 859)	{0   1}
:SBUS< n >:SENT:SIGNAl< s>:DISPlay { {0   OFF}   {1   ON} } (see page 860)	:SBUS< n >:SENT:SIGNAl< s>:DISPlay? (see page 860)	<s> ::= 1-6, in NR1 format. {0   1}
:SBUS< n >:SENT:SIGNAl< s>:LENGTH <length> (see page 861)	:SBUS< n >:SENT:SIGNAl< s>:LENGTH? (see page 861)	<s> ::= 1-6, in NR1 format. <length> ::= from 1-24, in NR1 format.
:SBUS< n >:SENT:SIGNAl< s>:MULTiplier <multiplier> (see page 863)	:SBUS< n >:SENT:SIGNAl< s>:MULTiplier? (see page 863)	<s> ::= 1-6, in NR1 format. <multiplier> ::= from 1-24, in NR3 format.
:SBUS< n >:SENT:SIGNAl< s>:OFFSet <offset> (see page 864)	:SBUS< n >:SENT:SIGNAl< s>:OFFSet? (see page 864)	<s> ::= 1-6, in NR1 format. <offset> ::= from 1-24, in NR3 format.

**Table 118:**:SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:SIGNAl<s>:ORDer <order> (see page 865)	:SBUS<n>:SENT:SIGNAl<s>:ORDer? (see page 865)	<s> ::= 1-6, in NR1 format. <order> ::= {MSNFFirst   LSNFirst}
:SBUS<n>:SENT:SIGNAl<s>:START <position> (see page 867)	:SBUS<n>:SENT:SIGNAl<s>:START? (see page 867)	<s> ::= 1-6, in NR1 format. <position> ::= from 0-23, in NR1 format.
:SBUS<n>:SENT:SOURce <source> (see page 869)	:SBUS<n>:SENT:SOURce? (see page 869)	<source> ::= {CHANnel<n>   DIGItal<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SENT:TOLeran ce <percent> (see page 871)	:SBUS<n>:SENT:TOLeran ce? (see page 871)	<percent> ::= from 3-30, in NR1 format.
:SBUS<n>:SENT:TRIGger <mode> (see page 872)	:SBUS<n>:SENT:TRIGger ? (see page 872)	<mode> ::= {SFCMessage   SSCMessage   FCData   SCMid   SCData   FCCerror   SCCerror   CRCerror   TOLerror   PPERror   SSPerror}
:SBUS<n>:SENT:TRIGger :FAST:DATA <string> (see page 874)	:SBUS<n>:SENT:TRIGger :FAST:DATA? (see page 874)	<string> ::= "nnnn..." where n ::= {0   1   X} <string> ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SENT:TRIGger :SLOW:DATA <data> (see page 875)	:SBUS<n>:SENT:TRIGger :SLOW:DATA? (see page 875)	<data> ::= when ILength = SHORT, from -1 (don't care) to 65535, in NR1 format. <data> ::= when ILength = LONG, from -1 (don't care) to 4095, in NR1 format.
:SBUS<n>:SENT:TRIGger :SLOW:ID <id> (see page 877)	:SBUS<n>:SENT:TRIGger :SLOW:ID? (see page 877)	<id> ::= when ILength = SHORT, from -1 (don't care) to 15, in NR1 format. <id> ::= when ILength = LONG, from -1 (don't care) to 255, in NR1 format.

**Table 118:**:SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:TRIGger :SLOW:ILENGTH <length> (see <a href="#">page 879</a> )	:SBUS<n>:SENT:TRIGger :SLOW:ILENGTH? (see <a href="#">page 879</a> )	<length> ::= {SHORT   LONG}
:SBUS<n>:SENT:TRIGger :TOLerance <percent> (see <a href="#">page 880</a> )	:SBUS<n>:SENT:TRIGger :TOLerance? (see <a href="#">page 880</a> )	<percent> ::= from 1-18, in NR1 format.

## :SBUS<n>:SENT:CLOCK

**N** (see [page 1334](#))

Command Syntax	<code>:SBUS&lt;n&gt;:SENT:CLOCK &lt;period&gt;</code>
	<code>&lt;period&gt;</code> ::= the nominal clock period (tick), from 1 us to 300 us, in NR 3 format.
	The :SBUS<n>:SENT:CLOCK command specifies the nominal clock period (tick), from 1 μs to 300 μs.
Query Syntax	<code>:SBUS&lt;n&gt;:SENT:CLOCK?</code>
	The :SBUS<n>:SENT:CLOCK? query returns the clock period setting.
Return Format	<code>&lt;period&gt;&lt;NL&gt;</code>
	<code>&lt;period&gt;</code> ::= the nominal clock period (tick), from 1 us to 300 us, in NR 3 format.
See Also	<ul style="list-style-type: none"> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:CRC</a>" on page 852</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:DISPLAY</a>" on page 853</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:FORMAT</a>" on page 855</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:IDLE</a>" on page 857</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:LENGTH</a>" on page 858</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:PPULSE</a>" on page 859</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:DISPLAY</a>" on page 860</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:LENGTH</a>" on page 861</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:MULTIPLIER</a>" on page 863</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:OFFSET</a>" on page 864</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:ORDER</a>" on page 865</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:START</a>" on page 867</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:SOURCe</a>" on page 869</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TOLERANCE</a>" on page 871</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER</a>" on page 872</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER:FAST:DATA</a>" on page 874</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER:SLOW:DATA</a>" on page 875</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER:SLOW:ID</a>" on page 877</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER:SLOW:ILENGTH</a>" on page 879</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SENT:TRIGGER:TOLERANCE</a>" on page 880</li> </ul>

## :SBUS<n>:SENT:CRC

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:CRC <format>`

`<format> ::= {LEGacy | RECommended}`

The :SBUS<n>:SENT:CRC command specifies the format of the CRC. Either Legacy (2008) or Recommended (2010).

Enhanced Serial Message CRCs are always calculated using the 2010 format, but for the Fast Channel Messages, and for Short Serial Message CRCs, this setting is used.

**Query Syntax**    `:SBUS<n>:SENT:CRC?`

The :SBUS<n>:SENT:CRC? query returns the CRC format setting.

**Return Format**    `<format><NL>`

`<format> ::= {LEG | REC}`

**See Also**

- [":SBUS<n>:SENT:CLOCK" on page 851](#)
- [":SBUS<n>:SENT:DISPlay" on page 853](#)
- [":SBUS<n>:SENT:FORMat" on page 855](#)
- [":SBUS<n>:SENT:IDLE" on page 857](#)
- [":SBUS<n>:SENT:LENGTH" on page 858](#)
- [":SBUS<n>:SENT:PPULse" on page 859](#)
- [":SBUS<n>:SENT:SIGNAl<s>:DISPlay" on page 860](#)
- [":SBUS<n>:SENT:SIGNAl<s>:LENGTH" on page 861](#)
- [":SBUS<n>:SENT:SIGNAl<s>:MULTiplier" on page 863](#)
- [":SBUS<n>:SENT:SIGNAl<s>:OFFSet" on page 864](#)
- [":SBUS<n>:SENT:SIGNAl<s>:ORDer" on page 865](#)
- [":SBUS<n>:SENT:SIGNAl<s>:START" on page 867](#)
- [":SBUS<n>:SENT:SOURce" on page 869](#)
- [":SBUS<n>:SENT:TOLerance" on page 871](#)
- [":SBUS<n>:SENT:TRIGger" on page 872](#)
- [":SBUS<n>:SENT:TRIGger:FAST:DATA" on page 874](#)
- [":SBUS<n>:SENT:TRIGger:SLOW:DATA" on page 875](#)
- [":SBUS<n>:SENT:TRIGger:SLOW:ID" on page 877](#)
- [":SBUS<n>:SENT:TRIGger:SLOW:ILENghth" on page 879](#)
- [":SBUS<n>:SENT:TRIGger:TOLerance" on page 880](#)

## :SBUS<n>:SENT:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:DISPlay <base>`  
`<base> ::= {HEX | DECimal | SYMBolic}`

The :SBUS<n>:SENT:DISPlay command specifies the number base used by the decoder. The chosen base is used for the data nibbles in Raw decode format, the defined Signals in the other formats, and for the data field of the Serial Messages.

This selection is used for both the Lister and the decode line displays.

When SYMBolic is selected, Fast Channel Signals display a calculated physical value based on the specified multiplier and offset:

- PhysicalValue = (Multiplier \* SignalValueAsUnsignedInteger) + Offset

When SYMBolic is selected, the CRC and Slow Channel information is displayed in hex.

**Query Syntax**    `:SBUS<n>:SENT:DISPlay?`

The :SBUS<n>:SENT:DISPlay? query returns the SENT decode number base setting.

**Return Format**    `<base><NL>`  
`<base> ::= {HEX | DEC | SYMB}`

**See Also**

- [":SBUS<n>:SENT:CLOCK" on page 851](#)
- [":SBUS<n>:SENT:CRC" on page 852](#)
- [":SBUS<n>:SENT:FORMAT" on page 855](#)
- [":SBUS<n>:SENT:IDLE" on page 857](#)
- [":SBUS<n>:SENT:LENGTH" on page 858](#)
- [":SBUS<n>:SENT:PPULSE" on page 859](#)
- [":SBUS<n>:SENT:SIGNAl<s>:DISPLAY" on page 860](#)
- [":SBUS<n>:SENT:SIGNAl<s>:LENGTH" on page 861](#)
- [":SBUS<n>:SENT:SIGNAl<s>:MULTIplier" on page 863](#)
- [":SBUS<n>:SENT:SIGNAl<s>:OFFSet" on page 864](#)
- [":SBUS<n>:SENT:SIGNAl<s>:ORDer" on page 865](#)
- [":SBUS<n>:SENT:SIGNAl<s>:START" on page 867](#)
- [":SBUS<n>:SENT:SOURce" on page 869](#)
- [":SBUS<n>:SENT:TOLERance" on page 871](#)
- [":SBUS<n>:SENT:TRIGger" on page 872](#)
- [":SBUS<n>:SENT:TRIGger:FAST:DATA" on page 874](#)

- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:FORMAT

**N** (see [page 1334](#))

### Command Syntax

```
:SBUS<n>:SENT:FORMAT <decode>
<decode> ::= {NIBBles | FSIGnal | FSSerial | FESErial | SSERial | ESEERial}
```

The :SBUS<n>:SENT:FORMAT command specifies the message decode/triggering format:

- NIBBles – displays the raw transmitted nibble values.
- FSIGnal – displays Fast Channel Message Signals.
- FSSerial – displays both Fast and Slow Messages (Short format) simultaneously.
- FESErial – displays both Fast and Slow Messages (Enhanced format) simultaneously.
- SSERial – displays Slow Channel Messages in Short format.
- ESEERial – displays Slow Channel Messages in Enhanced format.

This selection affects both decoding and triggering. The decode is affected both in how the system interprets the data, and what will be displayed. The trigger is affected in that the trigger hardware needs to be configured to trigger on serial messages correctly.

You can specify the nibble display order for Fast Channel Message Signals (see :SBUS<n>:SENT:SIGNAl<s>:ORDer). Raw transmitted nibble values are displayed in the order received.

Note that for the Slow Channel, the proper format, Short or Enhanced, must be chosen for proper decoding and triggering to occur.

Slow Channel Serial Messages are always displayed as defined by the SENT specification.

### Query Syntax

```
:SBUS<n>:SENT:FORMAT?
```

The :SBUS<n>:SENT:FORMAT? query returns the message decode/triggering format setting.

### Return Format

```
<decode><NL>
<decode> ::= {NIBB | FSIG | FSS | FES | SSER | ESER}
```

### See Also

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858

- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:IDLE

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:SENT:IDLE <state>
    <state> ::= {LOW | HIGH}
```

The :SBUS<n>:SENT:IDLE command specifies the idle state of the SENT bus.

**Query Syntax**

```
:SBUS<n>:SENT:IDLE?
```

The :SBUS<n>:SENT:IDLE? query returns the idle state setting.

**Return Format**

```
<state><NL>
    <state> ::= {LOW | HIGH}
```

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:LENGTH

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:LENGTH <#_nibbles>`

`<#_nibbles> ::= from 1-6, in NR1 format.`

The :SBUS<n>:SENT:LENGTH command specifies the number of nibbles in a SENT message, from 1 to 6.

**Query Syntax**    `:SBUS<n>:SENT:LENGTH?`

The :SBUS<n>:SENT:LENGTH? query returns the number of nibbles setting.

**Return Format**    `<#_nibbles><NL>`

`<#_nibbles> ::= from 1-6, in NR1 format.`

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLERance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

**:SBUS<n>:SENT:PPULse**

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:SENT:PPULse {{0 | OFF} | {1 | ON}}

The :SBUS<n>:SENT:PPULse command specifies whether the SENT messages are followed by a pause pulse.

**Query Syntax** :SBUS<n>:SENT:PPULse?

The :SBUS<n>:SENT:PPULse? query returns the pause pulse setting.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

**See Also** • [":SBUS<n>:SENT:CLOCK"](#) on page 851

• [":SBUS<n>:SENT:CRC"](#) on page 852

• [":SBUS<n>:SENT:DISPLAY"](#) on page 853

• [":SBUS<n>:SENT:FORMAT"](#) on page 855

• [":SBUS<n>:SENT:IDLE"](#) on page 857

• [":SBUS<n>:SENT:LENGTH"](#) on page 858

• [":SBUS<n>:SENT:SIGNAl<s>:DISPLAY"](#) on page 860

• [":SBUS<n>:SENT:SIGNAl<s>:LENGTH"](#) on page 861

• [":SBUS<n>:SENT:SIGNAl<s>:MULTIplier"](#) on page 863

• [":SBUS<n>:SENT:SIGNAl<s>:OFFSet"](#) on page 864

• [":SBUS<n>:SENT:SIGNAl<s>:ORDer"](#) on page 865

• [":SBUS<n>:SENT:SIGNAl<s>:START"](#) on page 867

• [":SBUS<n>:SENT:SOURce"](#) on page 869

• [":SBUS<n>:SENT:TOLERance"](#) on page 871

• [":SBUS<n>:SENT:TRIGGER"](#) on page 872

• [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 874

• [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 875

• [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 877

• [":SBUS<n>:SENT:TRIGger:SLOW:ILENghth"](#) on page 879

• [":SBUS<n>:SENT:TRIGger:TOLERance"](#) on page 880

## :SBUS<n>:SENT:SIGNAl<s>:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:SIGNAl<s>:DISPlay {{0 | OFF} | {1 | ON}}`  
`<s> ::= 1-6, in NR1 format.`

The :SBUS<n>:SENT:SIGNAl<s>:DISPlay command specifies whether the given signal is on or off.

**Query Syntax**    `:SBUS<n>:SENT:SIGNAl<s>:DISPlay?`

The :SBUS<n>:SENT:SIGNAl<s>:DISPlay? query returns the signal on/off setting.

**Return Format**    `<setting><NL>`

`<setting> ::= {0 | 1}`

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPlay](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:SIGNAl<s>:LENGTH

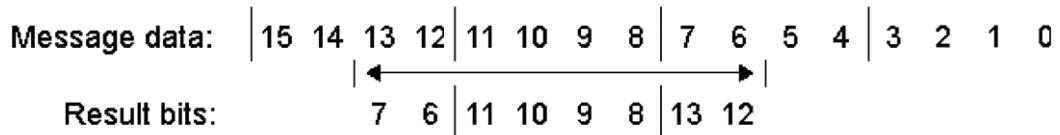
**N** (see page 1334)

**Command Syntax**    :SBUS<n>:SENT:SIGNAl<s>:LENGTH <length>  
                           <s> ::= 1-6, in NR1 format.  
                           <length> ::= from 1-24, in NR1 format.

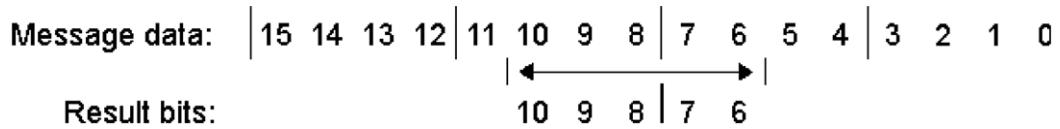
The :SBUS<n>:SENT:SIGNAl<s>:LENGTH command specifies the bit length of the signal being defined.

Fast Signal definition examples:

### Example 1: Start Bit # = 13, # of bits = 8, Nibble Order = LSN First



### Example 2: Start Bit # = 10, # of bits = 5, Nibble Order = MSN First



**Query Syntax**    :SBUS<n>:SENT:SIGNAl<s>:LENGTH?

The :SBUS<n>:SENT:SIGNAl<s>:LENGTH? query returns the signal bit length setting.

**Return Format**    <length><NL>  
                           <length> ::= from 1-24, in NR1 format.

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTIPLIER](#)" on page 863

- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:SIGNAl<s>:MULTiplier

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:SENT:SIGNAl<s>:MULTiplier <multiplier>
<s> ::= 1-6, in NR1 format.
<multiplier> ::= from 1-24, in NR3 format.
```

When the display mode setting is SYMBolic (see :SBUS<n>:SENT:DISPlay), the :SBUS<n>:SENT:SIGNAl<s>:MULTiplier command specifies the multiplier to be used in calculating a physical value displayed for a Fast Channel Signal.

- PhysicalValue = (Multiplier \* SignalValueAsUnsignedInteger) + Offset

**Query Syntax**

```
:SBUS<n>:SENT:SIGNAl<s>:MULTiplier?
```

The :SBUS<n>:SENT:SIGNAl<s>:MULTiplier? query returns the multiplier value for the Fast Channel Signal.

**Return Format**

```
<multiplier><NL>
<multiplier> ::= from 1-24, in NR3 format.
```

- See Also**
- [":SBUS<n>:SENT:CLOCK"](#) on page 851
  - [":SBUS<n>:SENT:CRC"](#) on page 852
  - [":SBUS<n>:SENT:DISPlay"](#) on page 853
  - [":SBUS<n>:SENT:FORMAT"](#) on page 855
  - [":SBUS<n>:SENT:IDLE"](#) on page 857
  - [":SBUS<n>:SENT:LENGTH"](#) on page 858
  - [":SBUS<n>:SENT:PPULse"](#) on page 859
  - [":SBUS<n>:SENT:SIGNAl<s>:DISPLAY"](#) on page 860
  - [":SBUS<n>:SENT:SIGNAl<s>:LENGTH"](#) on page 861
  - [":SBUS<n>:SENT:SIGNAl<s>:OFFSET"](#) on page 864
  - [":SBUS<n>:SENT:SIGNAl<s>:ORDer"](#) on page 865
  - [":SBUS<n>:SENT:SIGNAl<s>:START"](#) on page 867
  - [":SBUS<n>:SENT:SOURce"](#) on page 869
  - [":SBUS<n>:SENT:TOLerance"](#) on page 871
  - [":SBUS<n>:SENT:TRIGger"](#) on page 872
  - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 874
  - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 875
  - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 877
  - [":SBUS<n>:SENT:TRIGger:SLOW:ILENghth"](#) on page 879
  - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 880

## :SBUS<n>:SENT:SIGNAl<s>:OFFSet

**N** (see page 1334)

**Command Syntax**

```
:SBUS<n>:SENT:SIGNAl<s>:OFFSet <offset>
<s> ::= 1-6, in NR1 format.
<offset> ::= from 1-24, in NR3 format.
```

When the display mode setting is SYMBolic (see :SBUS<n>:SENT:DISPlay), the :SBUS<n>:SENT:SIGNAl<s>:OFFSet command is used in calculating a physical value displayed for the Fast Channel Signal:

- PhysicalValue = (Multiplier \* SignalValueAsUnsignedInteger) + Offset

**Query Syntax**

```
:SBUS<n>:SENT:SIGNAl<s>:OFFSet?
```

The :SBUS<n>:SENT:SIGNAl<s>:OFFSet? query returns the offset value for the Fast Channel Signal.

**Return Format**

```
<offset><NL>
<offset> ::= from 1-24, in NR3 format.
```

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLERance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGGER](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:SIGNAl<s>:ORDer

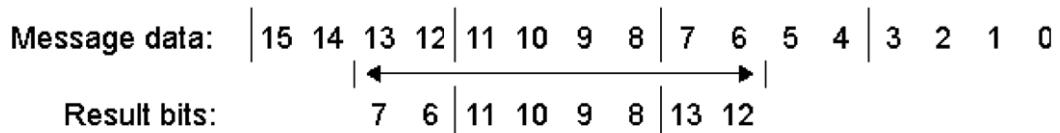
**N** (see page 1334)

**Command Syntax**    :SBUS<n>:SENT:SIGNAl<s>:ORDer <order>  
                           <s> ::= 1-6, in NR1 format.  
                           <order> ::= {MSNFirst | LSNFirst}

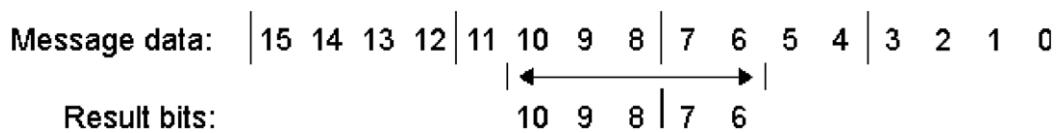
The :SBUS<n>:SENT:SIGNAl<s>:ORDer command specifies the nibble order of the signal being defined, either Most Significant Nibble first, or Least Significant Nibble first.

Fast Signal definition examples:

### Example 1: Start Bit # = 13, # of bits = 8, Nibble Order = LSN First



### Example 2: Start Bit # = 10, # of bits = 5, Nibble Order = MSN First



**Query Syntax**    :SBUS<n>:SENT:SIGNAl<s>:ORDer?

The :SBUS<n>:SENT:SIGNAl<s>:ORDer? query returns the nibble order setting.

**Return Format**    <order><NL>

<order> ::= {MSNF | LSNF}

**See Also**    · "[:SBUS<n>:SENT:CLOCK](#)" on page 851

· "[:SBUS<n>:SENT:CRC](#)" on page 852

· "[:SBUS<n>:SENT:DISPlay](#)" on page 853

· "[:SBUS<n>:SENT:FORMAT](#)" on page 855

· "[:SBUS<n>:SENT:IDLE](#)" on page 857

· "[:SBUS<n>:SENT:LENGTH](#)" on page 858

· "[:SBUS<n>:SENT:PPULse](#)" on page 859

· "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860

· "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861

- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:SIGNAl<s>:STARt

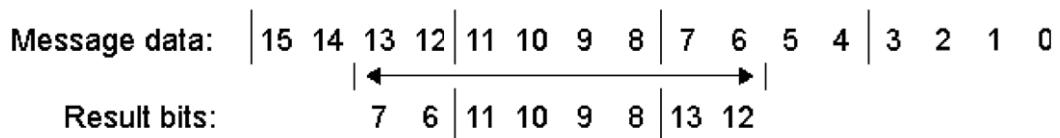
**N** (see page 1334)

**Command Syntax**    :SBUS<n>:SENT:SIGNAl<s>:STARt <position>  
                           <s> ::= 1-6, in NR1 format.  
                           <position> ::= from 0-23, in NR1 format.

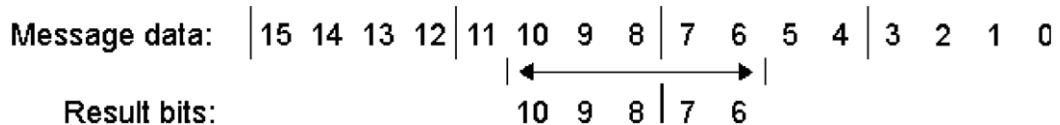
The :SBUS<n>:SENT:SIGNAl<s>:STARt command specifies the starting bit of the Fast Signal being defined.

Fast Signal definition examples:

### Example 1: Start Bit # = 13, # of bits = 8, Nibble Order = LSN First



### Example 2: Start Bit # = 10, # of bits = 5, Nibble Order = MSN First



**Query Syntax**    :SBUS<n>:SENT:SIGNAl<s>:STARt?

The :SBUS<n>:SENT:SIGNAl<s>:STARt? query returns the Fast Signal starting bit setting.

**Return Format**    <position><NL>  
                           <position> ::= from 0-23, in NR1 format.

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPlay](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861

- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:SOURce

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:SOURce <source>`

```
<source> ::= {CHANnel<n> | DIGital<d>}
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:SENT:SOURce command specifies the input channel for SENT decode and triggering.

**Query Syntax**    `:SBUS<n>:SENT:SOURce?`

The :SBUS<n>:SENT:SOURce? query returns the specified SENT input source.

**Return Format**    `<source><NL>`

```
<source> ::= {CHANnel<n> | DIGital<d>}
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:TOLERance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghT](#)" on page 879

- [":SBUS<n>:SENT:TRIGger:TOlerance" on page 880](#)

## :SBUS<n>:SENT:TOLerance

**N** (see page 1334)

**Command Syntax**    `:SBUS<n>:SENT:TOLerance <percent>`  
`<percent> ::= from 3-30, in NR1 format.`

The :SBUS<n>:SENT:TOLerance command specifies the tolerance for determining whether the sync pulse is valid. Valid values range from 3% to 30%.

**Query Syntax**    `:SBUS<n>:SENT:TOLerance?`

The :SBUS<n>:SENT:TOLerance? query returns the tolerance setting.

**Return Format**    `<percent><NL>`  
`<percent> ::= from 3-30, in NR1 format.`

- See Also**
- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
  - "[:SBUS<n>:SENT:CRC](#)" on page 852
  - "[:SBUS<n>:SENT:DISPlay](#)" on page 853
  - "[:SBUS<n>:SENT:FORMAT](#)" on page 855
  - "[:SBUS<n>:SENT:IDLE](#)" on page 857
  - "[:SBUS<n>:SENT:LENGTH](#)" on page 858
  - "[:SBUS<n>:SENT:PPULse](#)" on page 859
  - "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
  - "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
  - "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
  - "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
  - "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
  - "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
  - "[:SBUS<n>:SENT:SOURce](#)" on page 869
  - "[:SBUS<n>:SENT:TRIGger](#)" on page 872
  - "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
  - "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
  - "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
  - "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
  - "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:TRIGger

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:TRIGger <mode>`

```
<mode> ::= {SFCMessage | SSCMessage | FCDData | SCMid | SCData
             | TOLerror | FCCerror | SCCerror | CRCerror | PPERror | SSPerror}
```

The :SBUS<n>:SENT:TRIGger command specifies the SENT trigger mode:

- SFCMessage – triggers on the start of any Fast Channel Message (after 56 Synchronization/Calibration ticks).
- SSCMessage – trigger on the start of any Slow Channel Message.
- FCDData – triggers on a Fast Channel Message when the Status & Communication nibble and the data nibbles match the values entered using additional softkeys.
- SCMid – triggers when a Slow Channel Message ID matches the value entered using additional softkeys.
- SCData – triggers when a Slow Channel Message ID and Data field both match the values entered using additional softkeys.
- TOLerror – triggers when the sync pulse width varies from the nominal value by greater than the entered percentage.
- FCCerror – triggers on any Fast Channel Message CRC error.
- SCCerror – triggers on any Slow Channel Message CRC error.
- CRCerror – triggers on any CRC error, Fast or Slow.
- PPERror – triggers if a nibble is either too wide or too narrow (for example, data nibble < 12 (11.5) or > 27 (27.5) ticks wide). Sync, S&C, data, or checksum pulse periods are checked.
- SSPerror – triggers on a sync pulse whose width varies from the previous sync pulse's width by greater than 1/64 (1.5625%, as defined in the SENT specification).

**Query Syntax**    `:SBUS<n>:SENT:TRIGger?`

The :SBUS<n>:SENT:TRIGger? query returns the trigger mode setting.

**Return Format**    `<mode><NL>`

```
<mode> ::= {SFCM | SSCM | FCD | SCM | SCD | TOL | FCC | SCC | CRC
             | PPER | SSP}
```

### See Also

- "[":SBUS<n>:SENT:CLOCK](#)" on page 851
- "[":SBUS<n>:SENT:CRC](#)" on page 852
- "[":SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[":SBUS<n>:SENT:FORMAT](#)" on page 855
- "[":SBUS<n>:SENT:IDLE](#)" on page 857

- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTIplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:TRIGger:FAST:DATA

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:SENT:TRIGger:FAST:DATA &lt;string&gt;</code> <code>&lt;string&gt; ::= "nnnn..." where n ::= {0   1   x}</code> <code>&lt;string&gt; ::= "0xn..." where n ::= {0,...,9   A,...,F   x   \$}</code>
	The :SBUS<n>:SENT:TRIGger:FAST:DATA command specifies the status and data nibbles that will be triggered on when the FCData trigger mode is chosen.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:SENT:TRIGger:FAST:DATA?</code>
	The :SBUS<n>:SENT:TRIGger:FAST:DATA? query returns the fast channel data trigger setting.
<b>Return Format</b>	<code>&lt;string&gt;&lt;NL&gt;</code> <code>&lt;string&gt; ::= "nnnn..." where n ::= {0   1   x}</code> <code>&lt;string&gt; ::= "0xn..." where n ::= {0,...,9   A,...,F   x   \$}</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:CLOCK"</a> on page 851</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:CRC"</a> on page 852</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:DISPlay"</a> on page 853</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:FORMAT"</a> on page 855</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:IDLE"</a> on page 857</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:LENGTH"</a> on page 858</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:PPULse"</a> on page 859</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:DISPLAY"</a> on page 860</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:LENGTH"</a> on page 861</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:MULTiplier"</a> on page 863</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:OFFSet"</a> on page 864</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:ORDer"</a> on page 865</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SIGNAl&lt;s&gt;:START"</a> on page 867</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:SOURce"</a> on page 869</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TOLERance"</a> on page 871</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TRIGger"</a> on page 872</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TRIGger:slow:DATA"</a> on page 875</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TRIGger:slow:id"</a> on page 877</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TRIGger:slow:iLength"</a> on page 879</li> <li>· <a href="#">":SBUS&lt;n&gt;:SENT:TRIGger:TOLERance"</a> on page 880</li> </ul>

## :SBUS<n>:SENT:TRIGger:SLOW:DATA

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:SENT:TRIGger:SLOW:DATA <data>
<data> ::= when ILENgth = SHOrt, from -1 (don't care) to 65535, in NR1 f
ormat.
<data> ::= when ILENgth = LONG, from -1 (don't care) to 4095, in NR1 for
mat.
```

The :SBUS<n>:SENT:TRIGger:SLOW:DATA command specifies the data to trigger on for the Slow Channel Message ID and Data trigger mode.

**Query Syntax**

```
:SBUS<n>:SENT:TRIGger:SLOW:DATA?
```

The :SBUS<n>:SENT:TRIGger:SLOW:DATA? query returns the data value setting for the slow channel ID and data trigger.

**Return Format**

```
<data><NL>
<data> ::= when ILENgth = SHOrt, from -1 (don't care) to 65535, in NR1 f
ormat.
<data> ::= when ILENgth = LONG, from -1 (don't care) to 4095, in NR1 for
mat.
```

- See Also**
- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
  - "[:SBUS<n>:SENT:CRC](#)" on page 852
  - "[:SBUS<n>:SENT:DISPlay](#)" on page 853
  - "[:SBUS<n>:SENT:FORMAT](#)" on page 855
  - "[:SBUS<n>:SENT:IDLE](#)" on page 857
  - "[:SBUS<n>:SENT:LENGth](#)" on page 858
  - "[:SBUS<n>:SENT:PPULse](#)" on page 859
  - "[:SBUS<n>:SENT:SIGNaL<s>:DISPlay](#)" on page 860
  - "[:SBUS<n>:SENT:SIGNaL<s>:LENGth](#)" on page 861
  - "[:SBUS<n>:SENT:SIGNaL<s>:MULTiplier](#)" on page 863
  - "[:SBUS<n>:SENT:SIGNaL<s>:OFFSet](#)" on page 864
  - "[:SBUS<n>:SENT:SIGNaL<s>:ORDer](#)" on page 865
  - "[:SBUS<n>:SENT:SIGNaL<s>:START](#)" on page 867
  - "[:SBUS<n>:SENT:SOURce](#)" on page 869
  - "[:SBUS<n>:SENT:TOLerance](#)" on page 871
  - "[:SBUS<n>:SENT:TRIGger](#)" on page 872
  - "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
  - "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877

- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:TRIGger:SLOW:ID

**N** (see [page 1334](#))

**Command Syntax**

```
:SBUS<n>:SENT:TRIGger:SLOW:ID <id>
<id> ::= when ILENgth = SHOrt, from -1 (don't care) to 15, in NR1 format
.
<id> ::= when ILENgth = LONG, from -1 (don't care) to 255, in NR1 format
.
```

The :SBUS<n>:SENT:TRIGger:SLOW:ID command specifies the ID to trigger on for the "Slow Channel Message ID" and "Slow Channel Message ID & Data" trigger modes. The ID can be from -1 (don't care) to 255 (depending on the message length).

**Query Syntax**

```
:SBUS<n>:SENT:TRIGger:SLOW:ID?
```

The :SBUS<n>:SENT:TRIGger:SLOW:ID? query returns the slow channel ID setting.

**Return Format**

```
<id><NL>
<id> ::= when ILENgth = SHOrt, from -1 (don't care) to 15, in NR1 format
.
<id> ::= when ILENgth = LONG, from -1 (don't care) to 255, in NR1 format
.
```

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875

- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENgth](#)" on page 879
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:TRIGger:SLOW:ILENGTH

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:TRIGger:SLOW:ILENGTH <length>`  
`<length> ::= {SHORT | LONG}`

The :SBUS<n>:SENT:TRIGger:SLOW:ILENGTH command specifies the ID and data lengths for the Slow Message Enhanced messages. Either "SHORt" for the 4-bit ID, 16-bit data format, or "LONG" for the 8-bit ID, 12-bit data format.

**Query Syntax**    `:SBUS<n>:SENT:TRIGger:SLOW:ILENGTH?`

The :SBUS<n>:SENT:TRIGger:SLOW:ILENGTH? query returns the ID and data length setting.

**Return Format**    `<length><NL>`  
`<length> ::= {SHOR | LONG}`

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPlay](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULse](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPlay](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:TOLerance](#)" on page 880

## :SBUS<n>:SENT:TRIGger:TOLerance

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SENT:TRIGger:TOLerance <percent>`  
`<percent> ::= from 1-28, in NR1 format.`

The :SBUS<n>:SENT:TRIGger:TOLerance command specifies the tolerance variation that is considered a violation.

The trigger tolerance can be up to the :SBUS<n>:SENT:TOLerance setting minus two percent. For example, after sending ":SBUS1:SENT:TOLerance 20", the :SBUS1:SENT:TRIGger:TOLerance setting can be from 1 to 18.

**Query Syntax**    `:SBUS<n>:SENT:TRIGger:TOLerance?`

The :SBUS<n>:SENT:TRIGger:TOLerance? query returns tolerance variation percent setting.

**Return Format**    `<percent><NL>`  
`<percent> ::= from 1-28, in NR1 format.`

**See Also**

- "[:SBUS<n>:SENT:CLOCK](#)" on page 851
- "[:SBUS<n>:SENT:CRC](#)" on page 852
- "[:SBUS<n>:SENT:DISPLAY](#)" on page 853
- "[:SBUS<n>:SENT:FORMAT](#)" on page 855
- "[:SBUS<n>:SENT:IDLE](#)" on page 857
- "[:SBUS<n>:SENT:LENGTH](#)" on page 858
- "[:SBUS<n>:SENT:PPULSE](#)" on page 859
- "[:SBUS<n>:SENT:SIGNAl<s>:DISPLAY](#)" on page 860
- "[:SBUS<n>:SENT:SIGNAl<s>:LENGTH](#)" on page 861
- "[:SBUS<n>:SENT:SIGNAl<s>:MULTiplier](#)" on page 863
- "[:SBUS<n>:SENT:SIGNAl<s>:OFFSet](#)" on page 864
- "[:SBUS<n>:SENT:SIGNAl<s>:ORDer](#)" on page 865
- "[:SBUS<n>:SENT:SIGNAl<s>:START](#)" on page 867
- "[:SBUS<n>:SENT:SOURce](#)" on page 869
- "[:SBUS<n>:SENT:TOLerance](#)" on page 871
- "[:SBUS<n>:SENT:TRIGger](#)" on page 872
- "[:SBUS<n>:SENT:TRIGger:FAST:DATA](#)" on page 874
- "[:SBUS<n>:SENT:TRIGger:SLOW:DATA](#)" on page 875
- "[:SBUS<n>:SENT:TRIGger:SLOW:ID](#)" on page 877
- "[:SBUS<n>:SENT:TRIGger:SLOW:ILENghth](#)" on page 879

## :SBUS<n>:SPI Commands

**NOTE**

These commands are only valid when the low-speed IIC and SPI serial decode option (Option LSS) has been licensed.

**Table 119**:SBUS<n>:SPI Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SPI:BITorder <order> (see <a href="#">page 883</a> )	:SBUS<n>:SPI:BITorder ? (see <a href="#">page 883</a> )	<order> ::= {LSBFFirst   MSBFFirst}
:SBUS<n>:SPI:CLOCK:SLOPe <slope> (see <a href="#">page 884</a> )	:SBUS<n>:SPI:CLOCK:SLOPe? (see <a href="#">page 884</a> )	<slope> ::= {NEGative   POSitive}
:SBUS<n>:SPI:CLOCK:TI Meout <time_value> (see <a href="#">page 885</a> )	:SBUS<n>:SPI:CLOCK:TI Meout? (see <a href="#">page 885</a> )	<time_value> ::= time in seconds in NR3 format
:SBUS<n>:SPI:FRAMing <value> (see <a href="#">page 886</a> )	:SBUS<n>:SPI:FRAMing? (see <a href="#">page 886</a> )	<value> ::= {CHIPselect   {NCHipselect   NOTC}   TIMeout}
:SBUS<n>:SPI:SOURce:LOCK <source> (see <a href="#">page 887</a> )	:SBUS<n>:SPI:SOURce:LOCK? (see <a href="#">page 887</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:FRAMe <source> (see <a href="#">page 888</a> )	:SBUS<n>:SPI:SOURce:FRAMe? (see <a href="#">page 888</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:MIso <source> (see <a href="#">page 889</a> )	:SBUS<n>:SPI:SOURce:MIso? (see <a href="#">page 889</a> )	<value> ::= {CHANnel<n>   EXTERNAL} for the DSO models <value> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

**Table 119:**:SBUS<n>:SPI Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SPI:SOURce:MO SI <source> (see <a href="#">page 890</a> )	:SBUS<n>:SPI:SOURce:MO SI? (see <a href="#">page 890</a> )	<value> ::= {CHANnel<n>   EXTernal} for the DSO models <value> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:TRIGger: PATtern:MISO:DATA <string> (see <a href="#">page 891</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MISO:DATA? (see <a href="#">page 891</a> )	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SPI:TRIGger: PATtern:MISO:WIDTh <width> (see <a href="#">page 892</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MISO:WIDTh? (see <a href="#">page 892</a> )	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger: PATtern:MOStI:DATA <string> (see <a href="#">page 893</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MOStI:DATA? (see <a href="#">page 893</a> )	<string> ::= "nn...n" where n ::= {0   1   X   \$} <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$}
:SBUS<n>:SPI:TRIGger: PATtern:MOStI:WIDTh <width> (see <a href="#">page 894</a> )	:SBUS<n>:SPI:TRIGger: PATtern:MOStI:WIDTh? (see <a href="#">page 894</a> )	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger: TYPE <value> (see <a href="#">page 895</a> )	:SBUS<n>:SPI:TRIGger: TYPE? (see <a href="#">page 895</a> )	<value> ::= {MOStI   MISO}
:SBUS<n>:SPI:WIDTh <word_width> (see <a href="#">page 896</a> )	:SBUS<n>:SPI:WIDTh? (see <a href="#">page 896</a> )	<word_width> ::= integer 4-16 in NR1 format

**:SBUS<n>:SPI:BITorder****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:BITorder <order>`  
                  `<order> ::= {LSBFFirst | MSBFFirst}`

The :SBUS<n>:SPI:BITorder command selects the bit order, most significant bit first (MSB) or least significant bit first (LSB), used when displaying data in the serial decode waveform and in the Lister.

**Query Syntax**    `:SBUS<n>:SPI:BITorder?`

The :SBUS<n>:SPI:BITorder? query returns the current SPI decode bit order.

**Return Format**    `<order><NL>`  
                  `<order> ::= {LSBF | MSBF}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
                  • [":SBUS<n>:MODE" on page 727](#)  
                  • [":SBUS<n>:SPI Commands" on page 881](#)

**:SBUS<n>:SPI:CLOCK:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:CLOCK:SLOPe <slope>`  
                  `<slope> ::= {NEGative | POSitive}`

The :SBUS<n>:SPI:CLOCK:SLOPe command specifies the rising edge (POSitive) or falling edge (NEGative) of the SPI clock source that will clock in the data.

**Query Syntax**    `:SBUS<n>:SPI:CLOCK:SLOPe?`

The :SBUS<n>:SPI:CLOCK:SLOPe? query returns the current SPI clock source slope.

**Return Format**    `<slope><NL>`  
                  `<slope> ::= {NEG | POS}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":SBUS<n>:SPI:CLOCK:TIMEout" on page 885
- ":SBUS<n>:SPI:SOURce:CLOCK" on page 887

## :SBUS<n>:SPI:CLOCK:TIMEout

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:CLOCK:TIMEout &lt;time_value&gt;</code> <code>&lt;time_value&gt; ::= time in seconds in NR3 format</code>
	The :SBUS<n>:SPI:CLOCK:TIMEout command sets the SPI signal clock timeout resource in seconds from 100 ns to 10 s when the :SBUS<n>:SPI:FRAMing command is set to TIMEout. The timer is used to frame a signal by a clock timeout.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:CLOCK:TIMEout?</code>
	The :SBUS<n>:SPI:CLOCK:TIMEout? query returns current SPI clock timeout setting.
<b>Return Format</b>	<code>&lt;time value&gt;&lt;NL&gt;</code> <code>&lt;time_value&gt; ::= time in seconds in NR3 format</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:CLOCK:SLOPe</a>" on page 884</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:SOURce:CLOCK</a>" on page 887</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:FRAMing</a>" on page 886</li> </ul>

## :SBUS<n>:SPI:FRAMing

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:FRAMing <value>`

`<value> ::= {CHIPselect | {NCHipselect | NOTC} | TMeout}`

The :SBUS<n>:SPI:FRAMing command sets the SPI trigger framing value. If TMeout is selected, the timeout value is set by the :SBUS<n>:SPI:CLOCK:TMeout command.

### NOTE

The NOTC value is deprecated. It is the same as NCHipselect.

---

**Query Syntax**    `:SBUS<n>:SPI:FRAMing?`

The :SBUS<n>:SPI:FRAMing? query returns the current SPI framing value.

**Return Format**    `<value><NL>`

`<value> ::= {CHIP | NCH | TIM}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047

- ":TRIGger:MODE" on page 1056

- ":SBUS<n>:SPI:CLOCK:TMeout" on page 885

- ":SBUS<n>:SPI:SOURce:FRAMe" on page 888

## :SBUS<n>:SPI:SOURce:CLOCK

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:CLOCK &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   EXTERNAL} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGItal&lt;d&gt;} for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :SBUS<n>:SPI:SOURce:CLOCK command sets the source for the SPI serial clock.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:CLOCK?</code>
	The :SBUS<n>:SPI:SOURce:CLOCK? query returns the current source for the SPI serial clock.
<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:CLOCK:SLOPe</a>" on page 884</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:CLOCK:TIMEout</a>" on page 885</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:SOURce:FRAMe</a>" on page 888</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:SOURce:MOSI</a>" on page 890</li> <li>• "<a href="#">:SBUS&lt;n&gt;:SPI:SOURce:MISO</a>" on page 889</li> </ul>

## :SBUS<n>:SPI:SOURce:FRAMe

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:SOURce:FRAMe <source>`

```

<source> ::= {CHANnel<n> | EXTernal} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format

```

The :SBUS<n>:SPI:SOURce:FRAMe command sets the frame source when :SBUS<n>:SPI:FRAMing is set to CHIPselect or NOTChipselect.

**Query Syntax**    `:SBUS<n>:SPI:SOURce:FRAMe?`

The :SBUS<n>:SPI:SOURce:FRAMe? query returns the current frame source for the SPI serial frame.

**Return Format**    `<source><NL>`

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":SBUS<n>:SPI:SOURce:CLOCK](#)" on page 887
  - "[":SBUS<n>:SPI:SOURce:MOSI](#)" on page 890
  - "[":SBUS<n>:SPI:SOURce:MISO](#)" on page 889
  - "[":SBUS<n>:SPI:FRAMing](#)" on page 886

## :SBUS<n>:SPI:SOURce:MISO

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:MISO &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   EXternal} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;} for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :SBUS<n>:SPI:SOURce:MISO command sets the source for the SPI serial MISO data.

<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:MISO?</code>
	The :SBUS<n>:SPI:SOURce:MISO? query returns the current source for the SPI serial MISO data.

<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
----------------------	---------------------------------------

### See Also

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:SPI:SOURce:MOSI"](#) on page 890
- "[":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 887
- "[":SBUS<n>:SPI:SOURce:FRAMe"](#) on page 888
- "[":SBUS<n>:SPI:TRIGger:PATTern:MISO:DATA"](#) on page 891
- "[":SBUS<n>:SPI:TRIGger:PATTern:MOSI:DATA"](#) on page 893
- "[":SBUS<n>:SPI:TRIGger:PATTern:MISO:WIDTH"](#) on page 892
- "[":SBUS<n>:SPI:TRIGger:PATTern:MOSI:WIDTH"](#) on page 894

## :SBUS<n>:SPI:SOURce:MOSI

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:MOSI &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   EXternal} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;} for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :SBUS<n>:SPI:SOURce:MOSI command sets the source for the SPI serial MOSI data.

<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:SPI:SOURce:MOSI?</code>
	The :SBUS<n>:SPI:SOURce:MOSI? query returns the current source for the SPI serial MOSI data.

<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
----------------------	---------------------------------------

<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:SOURce:MISO</a>" on page 889</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:SOURce:CLOCK</a>" on page 887</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:SOURce:FRAMe</a>" on page 888</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:TRIGger:PATTern:MISO:DATA</a>" on page 891</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:TRIGger:PATTern:MOSI:DATA</a>" on page 893</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:TRIGger:PATTern:MISO:WIDTH</a>" on page 892</li> <li>· "<a href="#">":SBUS&lt;n&gt;:SPI:TRIGger:PATTern:MOSI:WIDTH</a>" on page 894</li> </ul>
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA <string>

```
<string> ::= "nn...n" where n ::= {0 | 1 | x | $}
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $}
```

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA command defines the SPI data pattern resource according to the string parameter. This pattern, along with the data width, control the data pattern searched for in the data stream.

If the string parameter starts with "0x", it is a hexadecimal string made up of hexadecimal and X (don't care) characters; otherwise, it is a binary string made up of 0, 1, and X (don't care) characters.

### NOTE

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh should be set before :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA.

**Query Syntax** :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA?

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA? query returns the current settings of the specified SPI data pattern resource in the binary string format.

**Return Format** <string><NL>

- See Also**
- ["Introduction to :TRIGger Commands" on page 1047](#)
  - [":SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh" on page 892](#)
  - [":SBUS<n>:SPI:SOURce:MISO" on page 889](#)

**:SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh <width>`  
`<width> ::= integer from 4 to 64 in NR1 format`

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh command sets the width of the SPI data pattern anywhere from 4 bits to 64 bits.

**NOTE**

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh should be set before :SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA.

**Query Syntax**    `:SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh?`

The :SBUS<n>:SPI:TRIGger:PATTERn:MISO:WIDTh? query returns the current SPI data pattern width setting.

**Return Format**    `<width><NL>`  
`<width> ::= integer from 4 to 64 in NR1 format`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:SPI:TRIGger:PATTERn:MISO:DATA](#)" on page 891
- "[:SBUS<n>:SPI:SOURce:MISO](#)" on page 889

## :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA <string>

```
<string> ::= "nn...n" where n ::= {0 | 1 | x | $}
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | x | $}
```

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA command defines the SPI data pattern resource according to the string parameter. This pattern, along with the data width, control the data pattern searched for in the data stream.

If the string parameter starts with "0x", it is a hexadecimal string made up of hexadecimal and X (don't care) characters; otherwise, it is a binary string made up of 0, 1, and X (don't care) characters.

### NOTE

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh should be set before :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA.

**Query Syntax** :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA?

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA? query returns the current settings of the specified SPI data pattern resource in the binary string format.

**Return Format** <string><NL>

- See Also**
- ["Introduction to :TRIGger Commands" on page 1047](#)
  - [":SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh" on page 894](#)
  - [":SBUS<n>:SPI:SOURce:MOSI" on page 890](#)

**:SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh <width>`  
`<width> ::= integer from 4 to 64 in NR1 format`

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh command sets the width of the SPI data pattern anywhere from 4 bits to 64 bits.

**NOTE**

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh should be set before :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA.

**Query Syntax**    `:SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh?`

The :SBUS<n>:SPI:TRIGger:PATTERn:MOSI:WIDTh? query returns the current SPI data pattern width setting.

**Return Format**    `<width><NL>`  
`<width> ::= integer from 4 to 64 in NR1 format`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:SPI:TRIGger:PATTERn:MOSI:DATA](#)" on page 893
- "[":SBUS<n>:SPI:SOURce:MOSI](#)" on page 890

## :SBUS<n>:SPI:TRIGger:TYPE

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:SPI:TRIGger:TYPE <value>`  
`<value> ::= {MOSI | MISO}`

The :SBUS<n>:SPI:TRIGger:TYPE command specifies whether the SPI trigger will be on the MOSI data or the MISO data.

When triggering on MOSI data, the data value is specified by the :SBUS<n>:SPI:TRIGger:PATTern:MOSI:DATA and :SBUS<n>:SPI:TRIGger:PATTern:MOSI:WIDTh commands.

When triggering on MISO data, the data value is specified by the :SBUS<n>:SPI:TRIGger:PATTern:MISO:DATA and :SBUS<n>:SPI:TRIGger:PATTern:MISO:WIDTh commands.

**Query Syntax**    `:SBUS<n>:SPI:TRIGger:TYPE?`

The :SBUS<n>:SPI:TRIGger:TYPE? query returns the current SPI trigger type setting.

**Return Format**    `<value><NL>`  
`<value> ::= {MOSI | MISO}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":SBUS<n>:SPI:SOURce:MOSI"](#) on page 890
- "[":SBUS<n>:SPI:SOURce:MISO"](#) on page 889
- "[":SBUS<n>:SPI:TRIGger:PATTern:MISO:DATA"](#) on page 891
- "[":SBUS<n>:SPI:TRIGger:PATTern:MOSI:DATA"](#) on page 893
- "[":SBUS<n>:SPI:TRIGger:PATTern:MISO:WIDTH"](#) on page 892
- "[":SBUS<n>:SPI:TRIGger:PATTern:MOSI:WIDTH"](#) on page 894
- "[":TRIGger:MODE"](#) on page 1056

**:SBUS<n>:SPI:WIDTH****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:SPI:WIDTH <word_width>`    `<word_width> ::= integer 4-16 in NR1 format`

The :SBUS<n>:SPI:WIDTH command determines the number of bits in a word of data for SPI.

**Query Syntax**    `:SBUS<n>:SPI:WIDTH?`

The :SBUS<n>:SPI:WIDTH? query returns the current SPI decode word width.

**Return Format**    `<word_width><NL>`    `<word_width> ::= integer 4-16 in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:SPI Commands" on page 881](#)

## :SBUS<n>:UART Commands

**NOTE**

These commands are only valid when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

**Table 120:**:SBUS<n>:UART Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:UART:BASE <base> (see page 900)	:SBUS<n>:UART:BASE? (see page 900)	<base> ::= {ASCII   BINARY   HEX}
:SBUS<n>:UART:BAUDrate e <baudrate> (see page 901)	:SBUS<n>:UART:BAUDrate e? (see page 901)	<baudrate> ::= integer from 100 to 8000000
:SBUS<n>:UART:BITorde r <bitorder> (see page 902)	:SBUS<n>:UART:BITorde r? (see page 902)	<bitorder> ::= {LSBFIRST   MSBFIRST}
n/a	:SBUS<n>:UART:COUNT:E RRor? (see page 903)	<frame_count> ::= integer in NR1 format
:SBUS<n>:UART:COUNT:R ESet (see page 904)	n/a	n/a
n/a	:SBUS<n>:UART:COUNT:R XFRAMES? (see page 905)	<frame_count> ::= integer in NR1 format
n/a	:SBUS<n>:UART:COUNT:T XFRAMES? (see page 906)	<frame_count> ::= integer in NR1 format
:SBUS<n>:UART:FRAMing <value> (see page 907)	:SBUS<n>:UART:FRAMing ? (see page 907)	<value> ::= {OFF   <decimal>   <nondecimal>} <decimal> ::= 8-bit integer from 0-255 (0x00-0xFF) <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary
:SBUS<n>:UART:PARity <parity> (see page 908)	:SBUS<n>:UART:PARity? (see page 908)	<parity> ::= {EVEN   ODD   NONE}
:SBUS<n>:UART:POLarit y <polarity> (see page 909)	:SBUS<n>:UART:POLarit y? (see page 909)	<polarity> ::= {HIGH   LOW}

**Table 120**:SBUS< n >:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS< n >:UART:SOURce: RX <source> (see <a href="#">page 910</a> )	:SBUS< n >:UART:SOURce: RX? (see <a href="#">page 910</a> )	<source> ::= {CHANnel< n >   EXTernal} for DSO models <source> ::= {CHANnel< n >   DIGItal< d >} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS< n >:UART:SOURce: TX <source> (see <a href="#">page 911</a> )	:SBUS< n >:UART:SOURce: TX? (see <a href="#">page 911</a> )	<source> ::= {CHANnel< n >   EXTernal} for DSO models <source> ::= {CHANnel< n >   DIGItal< d >} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS< n >:UART:TRIGger :BASE <base> (see <a href="#">page 912</a> )	:SBUS< n >:UART:TRIGger :BASE? (see <a href="#">page 912</a> )	<base> ::= {ASCii   HEX}
:SBUS< n >:UART:TRIGger :BURSt <value> (see <a href="#">page 913</a> )	:SBUS< n >:UART:TRIGger :BURSt? (see <a href="#">page 913</a> )	<value> ::= {OFF   1 to 4096 in NR1 format}
:SBUS< n >:UART:TRIGger :DATA <value> (see <a href="#">page 914</a> )	:SBUS< n >:UART:TRIGger :DATA? (see <a href="#">page 914</a> )	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, <binary>, or <quoted_string> format <hexadecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0   1} for binary <quoted_string> ::= any of the 128 valid 7-bit ASCII characters (or standard abbreviations)
:SBUS< n >:UART:TRIGger :IDLE <time_value> (see <a href="#">page 915</a> )	:SBUS< n >:UART:TRIGger :IDLE? (see <a href="#">page 915</a> )	<time_value> ::= time from 1 us to 10 s in NR3 format
:SBUS< n >:UART:TRIGger :QUALifier <value> (see <a href="#">page 916</a> )	:SBUS< n >:UART:TRIGger :QUALifier? (see <a href="#">page 916</a> )	<value> ::= {EQUAL   NOTEqual   GREaterthan   LESSthan}

**Table 120**:SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:UART:TRIGger :TYPE <value> (see <a href="#">page 917</a> )	:SBUS<n>:UART:TRIGger :TYPE? (see <a href="#">page 917</a> )	<value> ::= {RSTArt   RSTOP   RDATA   RD1   RD0   RDX   PARityerror   TSTAArt   TSTOP   TDATA   TD1   TD0   TDX}
:SBUS<n>:UART:WIDTh <width> (see <a href="#">page 918</a> )	:SBUS<n>:UART:WIDTh? (see <a href="#">page 918</a> )	<width> ::= {5   6   7   8   9}

**:SBUS<n>:UART:BASE****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:BASE <base>`  
                  `<base> ::= {ASCii | BINary | HEX}`

The :SBUS<n>:UART:BASE command determines the base to use for the UART decode and Lister display.

**Query Syntax**    `:SBUS<n>:UART:BASE?`

The :SBUS<n>:UART:BASE? query returns the current UART decode and Lister base setting.

**Return Format**    `<base><NL>`  
                  `<base> ::= {ASCii | BINary | HEX}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
                  • [":SBUS<n>:UART Commands" on page 897](#)

## :SBUS<n>:UART:BAUDrate

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:BAUDrate <baudrate>`

`<baudrate> ::= integer from 100 to 8000000`

The :SBUS<n>:UART:BAUDrate command selects the bit rate (in bps) for the serial decoder and/or trigger when in UART mode. The baud rate can be set from 100 b/s to 8 Mb/s.

If the baud rate you select does not match the system baud rate, false triggers may occur.

**Query Syntax**    `:SBUS<n>:UART:BAUDrate?`

The :SBUS<n>:UART:BAUDrate? query returns the current UART baud rate setting.

**Return Format**    `<baudrate><NL>`

`<baudrate> ::= integer from 100 to 8000000`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:MODE](#)" on page 1056
- "[":SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

**:SBUS<n>:UART:BITorder****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:BITorder <bitorder>`  
                      `<bitorder> ::= {LSBFFirst | MSBFFirst}`

The :SBUS<n>:UART:BITorder command specifies the order of transmission used by the physical Tx and Rx input signals for the serial decoder and/or trigger when in UART mode. LSBFirst sets the least significant bit of each message "byte" as transmitted first. MSBFirst sets the most significant bit as transmitted first.

**Query Syntax**    `:SBUS<n>:UART:BITorder?`

The :SBUS<n>:UART:BITorder? query returns the current UART bit order setting.

**Return Format**    `<bitorder><NL>`  
                      `<bitorder> ::= {LSBF | MSBF}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917
- "[:SBUS<n>:UART:SOURce:RX](#)" on page 910
- "[:SBUS<n>:UART:SOURce:TX](#)" on page 911

**:SBUS<n>:UART:COUNT:ERROr**

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:UART:COUNT:ERROr?

Returns the UART error frame count.

**Return Format** <frame\_count><NL>

<frame\_count> ::= integer in NR1 format

**Errors** • ["-241, Hardware missing" on page 1293](#)

**See Also** • [":SBUS<n>:UART:COUNT:RESet" on page 904](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:UART Commands" on page 897](#)

## :SBUS<n>:UART:COUNT:RESet

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:UART:COUNT:RESet

Resets the UART frame counters.

- Errors** · ["-241, Hardware missing" on page 1293](#)

- See Also** · [":SBUS<n>:UART:COUNT:ERRor" on page 903](#)  
· [":SBUS<n>:UART:COUNT:RXFRAMES" on page 905](#)  
· [":SBUS<n>:UART:COUNT:TXFRAMES" on page 906](#)  
· ["Introduction to :SBUS<n> Commands" on page 723](#)  
· [":SBUS<n>:MODE" on page 727](#)  
· [":SBUS<n>:UART Commands" on page 897](#)

**:SBUS<n>:UART:COUNT:RXFRAMES**

**N** (see [page 1334](#))

**Query Syntax** :SBUS<n>:UART:COUNT:RXFRAMES?

Returns the UART Rx frame count.

**Return Format** <frame\_count><NL>

<frame\_count> ::= integer in NR1 format

**Errors** • ["-241, Hardware missing"](#) on page 1293

**See Also** • [":SBUS<n>:UART:COUNT:RESET"](#) on page 904  
• ["Introduction to :SBUS<n> Commands"](#) on page 723  
• [":SBUS<n>:MODE"](#) on page 727  
• [":SBUS<n>:UART Commands"](#) on page 897

**:SBUS<n>:UART:COUNt:TXFRAMES****N** (see [page 1334](#))**Query Syntax**    `:SBUS<n>:UART:COUNt:TXFRAMES?`

Returns the UART Tx frame count.

**Return Format**    `<frame_count><NL>``<frame_count> ::= integer in NR1 format`**Errors**    • ["-241, Hardware missing" on page 1293](#)**See Also**    • [":SBUS<n>:UART:COUNt:RESet" on page 904](#)  
• ["Introduction to :SBUS<n> Commands" on page 723](#)  
• [":SBUS<n>:MODE" on page 727](#)  
• [":SBUS<n>:UART Commands" on page 897](#)

## :SBUS<n>:UART:FRAMing

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SBUS&lt;n&gt;:UART:FRAMing &lt;value&gt;</code>
	<code>&lt;value&gt; ::= {OFF   &lt;decimal&gt;   &lt;nondecimal&gt;}</code>
	<code>&lt;decimal&gt; ::= 8-bit integer in decimal from 0-255 (0x00-0xff)</code>
	<code>&lt;nondecimal&gt; ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal</code>
	<code>&lt;nondecimal&gt; ::= #Bnn...n where n ::= {0   1} for binary</code>
	The :SBUS<n>:UART:FRAMing command determines the byte value to use for framing (end of packet) or to turn off framing for UART decode.
<b>Query Syntax</b>	<code>:SBUS&lt;n&gt;:UART:FRAMing?</code>
	The :SBUS<n>:UART:FRAMing? query returns the current UART decode base setting.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code>
	<code>&lt;value&gt; ::= {OFF   &lt;decimal&gt;}</code>
	<code>&lt;decimal&gt; ::= 8-bit integer in decimal from 0-255</code>
<b>Errors</b>	<ul style="list-style-type: none"> <li>· <a href="#">"-241, Hardware missing"</a> on page 1293</li> </ul>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">"Introduction to :SBUS&lt;n&gt; Commands"</a> on page 723</li> <li>· <a href="#">":SBUS&lt;n&gt;:UART Commands"</a> on page 897</li> </ul>

**:SBUS<n>:UART:PARity****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:PARity <parity>`  
`<parity> ::= {EVEN | ODD | NONE}`

The :SBUS<n>:UART:PARity command selects the parity to be used with each message "byte" for the serial decoder and/or trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:PARity?`

The :SBUS<n>:UART:PARity? query returns the current UART parity setting.

**Return Format**    `<parity><NL>`  
`<parity> ::= {EVEN | ODD | NONE}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

## :SBUS<n>:UART:POLarity

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:POLarity <polarity>`  
`<polarity> ::= {HIGH | LOW}`

The :SBUS<n>:UART:POLarity command selects the polarity as idle low or idle high for the serial decoder and/or trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:POLarity?`

The :SBUS<n>:UART:POLarity? query returns the current UART polarity setting.

**Return Format**    `<polarity><NL>`  
`<polarity> ::= {HIGH | LOW}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":SBUS<n>:UART:TRIGger:TYPE" on page 917

**:SBUS<n>:UART:SOURce:RX****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:UART:SOURce:RX <source>``<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models``<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models``<n> ::= 1 to (# analog channels) in NR1 format``<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :SBUS<n>:UART:SOURce:RX command controls which signal is used as the Rx source by the serial decoder and/or trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:SOURce:RX?`

The :SBUS<n>:UART:SOURce:RX? query returns the current source for the UART Rx signal.

**Return Format**    `<source><NL>`

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:MODE](#)" on page 1056
  - "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917
  - "[:SBUS<n>:UART:BITorder](#)" on page 902

## :SBUS<n>:UART:SOURce:TX

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:UART:SOURce:TX <source>

```
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:UART:SOURce:TX command controls which signal is used as the Tx source by the serial decoder and/or trigger when in UART mode.

**Query Syntax** :SBUS<n>:UART:SOURce:TX?

The :SBUS<n>:UART:SOURce:TX? query returns the current source for the UART Tx signal.

**Return Format** <source><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:MODE](#)" on page 1056
  - "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917
  - "[:SBUS<n>:UART:BITorder](#)" on page 902

## :SBUS<n>:UART:TRIGger:BASE

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:TRIGger:BASE <base>`  
`<base> ::= {ASCII | HEX}`

The :SBUS<n>:UART:TRIGger:BASE command sets the front panel UART/RS232 trigger setup data selection option:

- ASCII – front panel data selection is from ASCII values.
- HEX – front panel data selection is from hexadecimal values.

The :SBUS<n>:UART:TRIGger:BASE setting does not affect the :SBUS<n>:UART:TRIGger:DATA command which can always set data values using ASCII or hexadecimal values.

### NOTE

The :SBUS<n>:UART:TRIGger:BASE command is independent of the :SBUS<n>:UART:BASE command which affects decode and Lister only.

**Query Syntax**    `:SBUS<n>:UART:TRIGger:BASE?`

The :SBUS<n>:UART:TRIGger:BASE? query returns the current UART base setting.

**Return Format**    `<base><NL>`

`<base> ::= {ASC | HEX}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:DATA](#)" on page 914

## :SBUS<n>:UART:TRIGger:BURSt

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:TRIGger:BURSt <value>`  
`<value> ::= {OFF | 1 to 4096 in NR1 format}`

The :SBUS<n>:UART:TRIGger:BURSt command selects the burst value (Nth frame after idle period) in the range 1 to 4096 or OFF, for the trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:TRIGger:BURSt?`

The :SBUS<n>:UART:TRIGger:BURSt? query returns the current UART trigger burst value.

**Return Format**    `<value><NL>`  
`<value> ::= {OFF | 1 to 4096 in NR1 format}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:IDLE](#)" on page 915
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

## :SBUS<n>:UART:TRIGger:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:TRIGger:DATA <value>`

```
<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal,
           <hexadecimal>, <binary>, or <quoted_string> format

<hexadecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal

<binary> ::= #Bnn...n where n ::= {0 | 1} for binary

<quoted_string> ::= any of the 128 valid 7-bit ASCII characters
                   (or standard abbreviations)
```

The :SBUS<n>:UART:TRIGger:DATA command selects the data byte value (0x00 to 0xFF) for the trigger QUALifier when in UART mode. The data value is used when one of the RD or TD trigger types is selected.

When entering an ASCII character via the quoted string, it must be one of the 128 valid characters (case-sensitive): "NUL", "SOH", "STX", "ETX", "EOT", "ENQ", "ACK", "BEL", "BS", "HT", "LF", "VT", "FF", "CR", "SO", "SI", "DLE", "DC1", "DC2", "DC3", "DC4", "NAK", "SYN", "ETB", "CAN", "EM", "SUB", "ESC", "FS", "GS", "RS", "US", "SP", "!", "\\", "#", "\$", "%", "&", "\\", "(", ")", "\*", "+", ",", "-", ".", "/", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", ":", ";", "<", "=", ">", "?", "@", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "[", "\\", "]", "^", "\_", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", "{", "|", "}", "~", or "DEL".

**Query Syntax**    `:SBUS<n>:UART:TRIGger:DATA?`

The :SBUS<n>:UART:TRIGger:DATA? query returns the current UART trigger data value.

**Return Format**    `<value><NL>`

```
<value> ::= 8-bit integer in decimal from 0-255
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:BASE](#)" on page 912
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

**:SBUS<n>:UART:TRIGger:IDLE**

**N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:TRIGger:IDLE <time_value>`

`<time_value> ::= time from 1 us to 10 s in NR3 format`

The :SBUS<n>:UART:TRIGger:IDLE command selects the value of the idle period for burst trigger in the range from 1 us to 10 s when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:TRIGger:IDLE?`

The :SBUS<n>:UART:TRIGger:IDLE? query returns the current UART trigger idle period time.

**Return Format**    `<time_value><NL>`

`<time_value> ::= time from 1 us to 10 s in NR3 format`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":SBUS<n>:UART:TRIGger:BURSt" on page 913
- ":SBUS<n>:UART:TRIGger:TYPE" on page 917

**:SBUS<n>:UART:TRIGger:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:SBUS<n>:UART:TRIGger:QUALifier <value>``<value> ::= {EQUAL | NOTEqual | GREaterthan | LESSthan}`

The :SBUS<n>:UART:TRIGger:QUALifier command selects the data qualifier when :TYPE is set to RDATa, RD1, RD0, RDX, TDATa, TD1, TD0, or TDX for the trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:TRIGger:QUALifier?`

The :SBUS<n>:UART:TRIGger:QUALifier? query returns the current UART trigger qualifier.

**Return Format**    `<value><NL>``<value> ::= {EQU | NOT | GRE | LESS}`**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

## :SBUS<n>:UART:TRIGger:TYPE

**N** (see [page 1334](#))

**Command Syntax** :SBUS<n>:UART:TRIGger:TYPE <value>

```
<value> ::= {RSTA | RSTO | RDAT | RD1 | RD0 | RDX | PARityerror
             | TSTA | TSTO | TDAT | TD1 | TD0 | TDX}
```

The :SBUS<n>:UART:TRIGger:TYPE command selects the UART trigger type.

When one of the RD or TD types is selected, the :SBUS<n>:UART:TRIGger:DATA and :SBUS<n>:UART:TRIGger:QUALifier commands are used to specify the data value and comparison operator.

The RD1, RD0, RDX, TD1, TD0, and TDX types (for triggering on data and alert bit values) are only valid when a 9-bit width has been selected.

**Query Syntax** :SBUS<n>:UART:TRIGger:TYPE?

The :SBUS<n>:UART:TRIGger:TYPE? query returns the current UART trigger data value.

**Return Format** <value><NL>

```
<value> ::= {RSTA | RSTO | RDAT | RD1 | RD0 | RDX | PAR | TSTA |
             TSTO | TDAT | TD1 | TD0 | TDX}
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:DATA](#)" on page 914
- "[:SBUS<n>:UART:TRIGger:QUALifier](#)" on page 916
- "[:SBUS<n>:UART:WIDTh](#)" on page 918

**:SBUS<n>:UART:WIDTh****N** (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:UART:WIDTh <width>`  
`<width> ::= {5 | 6 | 7 | 8 | 9}`

The :SBUS<n>:UART:WIDTh command determines the number of bits (5–9) for each message "byte" for the serial decoder and/or trigger when in UART mode.

**Query Syntax**    `:SBUS<n>:UART:WIDTh?`

The :SBUS<n>:UART:WIDTh? query returns the current UART width setting.

**Return Format**    `<width><NL>`  
`<width> ::= {5 | 6 | 7 | 8 | 9}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:SBUS<n>:UART:TRIGger:TYPE](#)" on page 917

## 30 :SEARch Commands

Control the event search modes and parameters for each search type. See:

- "[General :SEARch Commands](#)" on page 920
- "[:SEARch:EDGE Commands](#)" on page 925
- "[:SEARch:GLITch Commands](#)" on page 928 (Pulse Width search)
- "[:SEARch:PEAK Commands](#)" on page 935
- "[:SEARch:RUNT Commands](#)" on page 940
- "[:SEARch:TRANSition Commands](#)" on page 945
- "[:SEARch:SERial:A429 Commands](#)" on page 950
- "[:SEARch:SERial:CAN Commands](#)" on page 956
- "[:SEARch:SERial:FLEXray Commands](#)" on page 966
- "[:SEARch:SERial:I2S Commands](#)" on page 972
- "[:SEARch:SERial:IIC Commands](#)" on page 978
- "[:SEARch:SERial:LIN Commands](#)" on page 985
- "[:SEARch:SERial:M1553 Commands](#)" on page 994
- "[:SEARch:SERial:SENT Commands](#)" on page 998
- "[:SEARch:SERial:SPI Commands](#)" on page 1003
- "[:SEARch:SERial:UART Commands](#)" on page 1007

## General :SEARch Commands

**Table 121** General :SEARch Commands Summary

Command	Query	Options and Query Returns
n/a	:SEARch:COUNT? (see page 921)	<count> ::= an integer count value
:SEARch:EVENT <event_number> (see page 922)	:SEARch:EVENT? (see page 922)	<event_number> ::= the integer number of a found search event
:SEARch:MODE <value> (see page 923)	:SEARch:MODE? (see page 923)	<value> ::= {EDGE   GLITch   RUNT   TRANSition   SERial{1   2}   PEAK}
:SEARch:STATE <value> (see page 924)	:SEARch:STATE? (see page 924)	<value> ::= {{0   OFF}   {1   ON}}

**:SEARch:COUNT**

**N** (see [page 1334](#))

**Query Syntax** :SEARch:COUNT?

The :SEARch:COUNT? query returns the number of search events found.

**Return Format** <count><NL>

<count> ::= an integer count value

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:EVENT"](#) on page 922
  - [":SEARch:STATe"](#) on page 924
  - [":SEARch:MODE"](#) on page 923

**:SEARch:EVENT****N** (see [page 1334](#))**Command Syntax**    `:SEARch:EVENT <event_number>``<event_number> ::= the integer number of a found search event`

The :SEARch:EVENT command navigates to a found search event.

If the :SEARch:STATe is ON, the horizontal position is changed so that the specified event is located at the time reference.

**Query Syntax**    `:SEARch:EVENT?`

The :SEARch:EVENT? query returns the currently selected event number.

**Return Format**    `<event_number><NL>``<event_number> ::= the integer number of a found search event`

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:COUNt"](#) on page 921
  - [":SEARch:STATe"](#) on page 924
  - [":SEARch:MODE"](#) on page 923

## :SEARch:MODE

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SEARch:MODE &lt;value&gt;</code> <code>&lt;value&gt; ::= {EDGE   GLITch   RUNT   TRANSition   SERial{1   2}   PEAK}</code>
	The :SEARch:MODE command selects the search mode.
	The command is only valid when the :SEARch:STATe is ON.
<b>Query Syntax</b>	<code>:SEARch:MODE?</code>
	The :SEARch:MODE? query returns the currently selected mode or OFF if the :SEARch:STATe is OFF.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code> <code>&lt;value&gt; ::= {EDGE   GLIT   RUNT   TRAN   SER{1   2}   PEAK   OFF}</code>
<b>See Also</b>	<ul style="list-style-type: none"><li><a href="#">Chapter 30</a>, “:SEARch Commands,” starting on page 919</li><li><a href="#">":SEARch:STATe"</a> on page 924</li><li><a href="#">":SEARch:COUNt"</a> on page 921</li><li><a href="#">":SEARch:EVENT"</a> on page 922</li></ul>

## :SEARch:STATe

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:STATe <value>`

`<value> ::= {{0 | OFF} | {1 | ON}}`

The :SEARch:STATe command enables or disables the search feature.

**Query Syntax**    `:SEARch:STATe?`

The :SEARch:STATe? query returns the current setting.

**Return Format**    `<value><NL>`

`<value> ::= {0 | 1}`

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:MODE"](#) on page 923
  - [":SEARch:COUNt"](#) on page 921
  - [":SEARch:EVENT"](#) on page 922

## :SEARch:EDGE Commands

**Table 122** :SEARch:EDGE Commands Summary

Command	Query	Options and Query Returns
:SEARch:EDGE:SLOPe <slope> (see <a href="#">page 926</a> )	:SEARch:EDGE:SLOPe? (see <a href="#">page 926</a> )	<slope> ::= {POSitive   NEGative   EITHer}
:SEARch:EDGE:SOURce <source> (see <a href="#">page 927</a> )	:SEARch:EDGE:SOURce? (see <a href="#">page 927</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

**:SEARch:EDGE:SLOPe****N** (see [page 1334](#))**Command Syntax**    `:SEARch:EDGE:SLOPe <slope>``<slope> ::= {NEGative | POSitive | EITHer}`

The :SEARch:EDGE:SLOPe command specifies the slope of the edge for the search.

**Query Syntax**    `:SEARch:EDGE:SLOPe?`

The :SEARch:EDGE:SLOPe? query returns the current slope setting.

**Return Format**    `<slope><NL>``<slope> ::= {NEG | POS | EITH}`**See Also**    • [Chapter 30](#), “:SEARch Commands,” starting on page 919

**:SEARch:EDGE:SOURce****N** (see [page 1334](#))**Command Syntax**    `:SEARch:EDGE:SOURce <source>`    `<source> ::= CHANnel<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :SEARch:EDGE:SOURce command selects the channel on which to search for edges.

**Query Syntax**    `:SEARch:EDGE:SOURce?`

The :SEARch:EDGE:SOURce? query returns the current source.

**Return Format**    `<source><NL>`    `<source> ::= CHAN<n>`**See Also**    • [Chapter 30](#), “:SEARch Commands,” starting on page 919

## :SEARch:GLITch Commands

**Table 123**:SEARch:GLITch Commands Summary

Command	Query	Options and Query Returns
:SEARch:GLITch:GREaterthan <greater_than_time>[suffix] (see <a href="#">page 929</a> )	:SEARch:GLITch:GREaterthan? (see <a href="#">page 929</a> )	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:LESSthan <less_than_time>[suffix] (see <a href="#">page 930</a> )	:SEARch:GLITch:LESSthan? (see <a href="#">page 930</a> )	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:POLarity <polarity> (see <a href="#">page 931</a> )	:SEARch:GLITch:POLarity? (see <a href="#">page 931</a> )	<polarity> ::= {POSitive   NEGative}
:SEARch:GLITch:QUALifier <qualifier> (see <a href="#">page 932</a> )	:SEARch:GLITch:QUALifier? (see <a href="#">page 932</a> )	<qualifier> ::= {GREaterthan   LESSthan   RANGE}
:SEARch:GLITch:RANGE <less_than_time>[suffix], <greater_than_time>[suffix] (see <a href="#">page 933</a> )	:SEARch:GLITch:RANGE? (see <a href="#">page 933</a> )	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:SEARch:GLITch:SOURce <source> (see <a href="#">page 934</a> )	:SEARch:GLITch:SOURce? (see <a href="#">page 934</a> )	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format

## :SEARch:GLITch:GREaterthan

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:GLITch:GREaterthan <greater_than_time>[<suffix>]`

`<greater_than_time> ::= floating-point number in NR3 format`

`<suffix> ::= {s | ms | us | ns | ps}`

The :SEARch:GLITch:GREaterthan command sets the minimum pulse width duration for the selected :SEARch:GLITch:SOURce.

**Query Syntax**    `:SEARch:GLITch:GREaterthan?`

The :SEARch:GLITch:GREaterthan? query returns the minimum pulse width duration time for :SEARch:GLITch:SOURce.

**Return Format**    `<greater_than_time><NL>`

`<greater_than_time> ::= floating-point number in NR3 format.`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:GLITch:SOURce"](#) on page 934
- [":SEARch:GLITch:QUALifier"](#) on page 932
- [":SEARch:MODE"](#) on page 923

**:SEARch:GLITch:LESSthan****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:GLITch:LESSthan <less_than_time>[<suffix>]`  
`<less_than_time> ::= floating-point number in NR3 format`  
`<suffix> ::= {s | ms | us | ns | ps}`

The :SEARch:GLITch:LESSthan command sets the maximum pulse width duration for the selected :SEARch:GLITch:SOURce.

**Query Syntax**    `:SEARch:GLITch:LESSthan?`

The :SEARch:GLITch:LESSthan? query returns the pulse width duration time for :SEARch:GLITch:SOURce.

**Return Format**    `<less_than_time><NL>`  
`<less_than_time> ::= floating-point number in NR3 format.`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:GLITch:SOURce"](#) on page 934
- [":SEARch:GLITch:QUALifier"](#) on page 932
- [":SEARch:MODE"](#) on page 923

**:SEARch:GLITch:POLarity****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:GLITch:POLarity <polarity>`  
`<polarity> ::= {POSitive | NEGative}`

The :SEARch:GLITch:POLarity command sets the polarity for the glitch (pulse width) search.

**Query Syntax**    `:SEARch:GLITch:POLarity?`

The :SEARch:GLITch:POLarity? query returns the current polarity setting for the glitch (pulse width) search.

**Return Format**    `<polarity><NL>`  
`<polarity> ::= {POS | NEG}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:MODE"](#) on page 923
- [":SEARch:GLITch:SOURce"](#) on page 934

**:SEARch:GLITch:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:SEARch:GLITch:QUALifier <operator>``<operator> ::= {GREaterthan | LESSthan | RANGE}`

This command sets the mode of operation of the glitch (pulse width) search. The oscilloscope can search for a pulse width that is greater than a time value, less than a time value, or within a range of time values.

**Query Syntax**    `:SEARch:GLITch:QUALifier?`

The `:SEARch:GLITch:QUALifier?` query returns the glitch (pulse width) qualifier.

**Return Format**    `<operator><NL>``<operator> ::= {GRE | LESS | RANG}`

- See Also**
- [Chapter 30](#), “`:SEARch` Commands,” starting on page 919
  - [`":SEARch:GLITch:SOURce"`](#) on page 934
  - [`":SEARch:MODE"`](#) on page 923

## :SEARch:GLITch:RANGE

**N** (see [page 1334](#))

**Command Syntax**

```
:SEARch:GLITch:RANGE <less_than_time>[suffix],  
                      <greater_than_time>[suffix]  
  
<less_than_time> ::= (15 ns - 10 seconds) in NR3 format  
  
<greater_than_time> ::= (10 ns - 9.99 seconds) in NR3 format  
  
[suffix] ::= {s | ms | us | ns | ps}
```

The :SEARch:GLITch:RANGE command sets the pulse width duration for the selected :SEARch:GLITch:SOURce. You can enter the parameters in any order – the smaller value becomes the <greater\_than\_time> and the larger value becomes the <less\_than\_time>.

**Query Syntax**

```
:SEARch:GLITch:RANGE?
```

The :SEARch:GLITch:RANGE? query returns the pulse width duration time for :SEARch:GLITch:SOURce.

**Return Format**

```
<less_than_time>, <greater_than_time><NL>
```

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:GLITch:SOURce"](#) on page 934
- [":SEARch:GLITch:QUALifier"](#) on page 932
- [":SEARch:MODE"](#) on page 923

**:SEARch:GLITch:SOURce****N** (see [page 1334](#))**Command Syntax**    `:SEARch:GLITch:SOURce <source>`    `<source> ::= CHANnel<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :SEARch:GLITch:SOURce command selects the channel on which to search for glitches (pulse widths).

**Query Syntax**    `:SEARch:GLITch:SOURce?`

The :SEARch:GLITch:SOURce? query returns the current pulse width source.

If all channels are off, the query returns "NONE."

**Return Format**    `<source><NL>`

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:MODE"](#) on page 923
  - [":SEARch:GLITch:POLarity"](#) on page 931
  - [":SEARch:GLITch:QUALifier"](#) on page 932
  - [":SEARch:GLITch:RANGE"](#) on page 933

## :SEARch:PEAK Commands

**Table 124:** :SEARch:PEAK Commands Summary

Command	Query	Options and Query Returns
:SEARch:PEAK:EXCursion <delta_level> (see <a href="#">page 936</a> )	:SEARch:PEAK:EXCursion? (see <a href="#">page 936</a> )	<delta_level> ::= required change in level to be recognized as a peak, in NR3 format.
:SEARch:PEAK:NPEaks <number> (see <a href="#">page 937</a> )	:SEARch:PEAK:NPEaks? (see <a href="#">page 937</a> )	<number> ::= max number of peaks to find, 1-11 in NR1 format.
:SEARch:PEAK:SOURCE <source> (see <a href="#">page 938</a> )	:SEARch:PEAK:SOURce? (see <a href="#">page 938</a> )	<source> ::= {FUNCTION<m>   MATH<m>   FFT} (must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format
:SEARch:PEAK:THReShold <level> (see <a href="#">page 939</a> )	:SEARch:PEAK:THReShold? (see <a href="#">page 939</a> )	<level> ::= necessary level to be considered a peak, in NR3 format.

**:SEARch:PEAK:EXCursion****N** (see [page 1334](#))**Command Syntax**    `:SEARch:PEAK:EXCursion <delta_level>`

`<delta_level>` ::= required change in level to be recognized as a peak, in NR3 format.

The :SEARch:PEAK:EXCursion command specifies the change in level that must occur (in other words, hysteresis) to be recognized as a peak.

The threshold level units are specified by the :FFT:VTYPe or :FUNCTION<m>[:FFT]:VTYPe command.

**Query Syntax**    `:SEARch:PEAK:EXCursion?`

The :SEARch:PEAK:EXCursion? query returns the specified excursion delta level value.

**Return Format**    `<delta_level><NL>`

`<delta_level>` ::= in NR3 format.

- See Also**

- "[:FFT:VTYPe](#)" on page 376
- "[:FUNCTION<m>\[:FFT\]:VTYPe](#)" on page 397
- "[:SEARch:PEAK:NPEaks](#)" on page 937
- "[:SEARch:PEAK:SOURce](#)" on page 938
- "[:SEARch:PEAK:THreshold](#)" on page 939

**:SEARch:PEAK:NPEaks**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:PEAK:NPEaks <number>`

`<number>` ::= max number of peaks to find, 1-11 in NR1 format.

The :SEARch:PEAK:NPEaks command specifies the maximum number of FFT peaks to find. This number can be from 1 to 11.

**Query Syntax**    `:SEARch:PEAK:NPEaks?`

The :SEARch:PEAK:NPEaks? query returns the specified maximum number of FFT peaks to find.

**Return Format**    `<number><NL>`

`<number>` ::= in NR1 format.

**See Also**

- [":SEARch:PEAK:EXCursion"](#) on page 936
- [":SEARch:PEAK:SOURce"](#) on page 938
- [":SEARch:PEAK:THreshold"](#) on page 939

**:SEARch:PEAK:SOURce****N** (see [page 1334](#))**Command Syntax**    `:SEARch:PEAK:SOURCE <source>`

```
<source> ::= {FUNCTION<m> | MATH<m> | FFT} (source must be  
an FFT waveform)
```

```
<m> ::= 1 to (# math functions) in NR1 format
```

The :SEARch:PEAK:SOURce command selects the FFT math function waveform to search.

**Query Syntax**    `:SEARch:PEAK:SOURCE?`

The :SEARch:PEAK:SOURce? query returns the FFT math function waveform that is being searched.

**Return Format**    `<source><NL>`

```
<source> ::= {FUNC<m> | FFT} (must be FFT)
```

```
<m> ::= 1 to (# math functions) in NR1 format
```

**See Also**

- "[:SEARch:PEAK:EXCursion](#)" on page 936
- "[:SEARch:PEAK:NPEaks](#)" on page 937
- "[:SEARch:PEAK:THreshold](#)" on page 939

## :SEARch:PEAK:THreshold

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:PEAK:THreshold <level>`

`<level>` ::= necessary level to be considered a peak, in NR3 format.

The :SEARch:PEAK:THreshold command specifies the threshold level necessary to be considered a peak.

The threshold level units are specified by the :FFT:VTYPe or :FUNCTION<m>[:FFT]:VTYPe command.

**Query Syntax**    `:SEARch:PEAK:THreshold?`

The :SEARch:PEAK:THreshold? query returns the specified threshold level for FFT peak search.

**Return Format**    `<level><NL>`

`<level>` ::= in NR3 format.

- See Also**
- "[:FFT:VTYPe](#)" on page 376
  - "[:FUNCTION<m>\[:FFT\]:VTYPe](#)" on page 397
  - "[:SEARch:PEAK:EXCusion](#)" on page 936
  - "[:SEARch:PEAK:NPEaks](#)" on page 937
  - "[:SEARch:PEAK:SOURce](#)" on page 938

## :SEARch:RUNT Commands

**Table 125:** :SEARch:RUNT Commands Summary

Command	Query	Options and Query Returns
:SEARch:RUNT:POLarity <polarity> (see <a href="#">page 941</a> )	:SEARch:RUNT:POLarity? ? (see <a href="#">page 941</a> )	<polarity> ::= {POSitive   NEGative   EITHer}
:SEARch:RUNT:QUALifie r <qualifier> (see <a href="#">page 942</a> )	:SEARch:RUNT:QUALifie r? ? (see <a href="#">page 942</a> )	<qualifier> ::= {GREaterthan   LESSthan   NONE}
:SEARch:RUNT:SOURCE <source> (see <a href="#">page 943</a> )	:SEARch:RUNT:SOURce? (see <a href="#">page 943</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARch:RUNT:TIME <time>[suffix] (see <a href="#">page 944</a> )	:SEARch:RUNT:TIME? (see <a href="#">page 944</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

## :SEARch:RUNT:POLarity

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:RUNT:POLarity <slope>`

`<polarity> ::= {POSITIVE | NEGATIVE | EITHER}`

The :SEARch:RUNT:POLarity command sets the polarity for the runt search.

**Query Syntax**    `:SEARch:RUNT:POLarity?`

The :SEARch:RUNT:POLarity? query returns the currently set runt polarity.

**Return Format**    `<slope><NL>`

`<polarity> ::= {POS | NEG | EITHER}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919

- [“:SEARch:MODE”](#) on page 923

- [“:SEARch:RUNT:SOURce”](#) on page 943

**:SEARch:RUNT:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:SEARch:RUNT:QUALifier <qualifier>``<qualifier> ::= {GREaterthan | LESSthan | NONE}`

The :SEARch:RUNT:QUALifier command specifies whether to search for a runt that is greater than a time value, less than a time value, or any time value.

**Query Syntax**    `:SEARch:RUNT:QUALifier?`

The :SEARch:RUNT:QUALifier? query returns the current runt search qualifier.

**Return Format**    `<qualifier><NL>``<qualifier> ::= {GRE | LESS | NONE}`**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [“:SEARch:MODE”](#) on page 923

**:SEARch:RUNT:SOURce****N** (see [page 1334](#))**Command Syntax**    `:SEARch:RUNT:SOURce <source>`    `<source> ::= CHANnel<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :SEARch:RUNT:SOURce command selects the channel on which to search for the runt pulse.

**Query Syntax**    `:SEARch:RUNT:SOURce?`

The :SEARch:RUNT:SOURce? query returns the current runt search source.

**Return Format**    `<source><NL>`    `<source> ::= CHAN<n>`**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:RUNT:POLarity"](#) on page 941

**:SEARch:RUNT:TIME****N** (see [page 1334](#))**Command Syntax**    `:SEARch:RUNT:TIME <time>[suffix]`    `<time> ::= floating-point number in NR3 format`    `[suffix] ::= {s | ms | us | ns | ps}`

When searching for runt pulses whose widths are greater than or less than a time (see :SEARch:RUNT:QUALifier), the :SEARch:RUNT:TIME command specifies the time value.

**Query Syntax**    `:SEARch:RUNT:TIME?`

The :SEARch:RUNT:TIME? query returns the currently specified runt time value.

**Return Format**    `<time><NL>`    `<time> ::= floating-point number in NR3 format`**See Also**

- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
- [“:SEARch:RUNT:QUALifier” on page 942](#)

## :SEARch:TRANSition Commands

**Table 126**:SEARch:TRANSition Commands Summary

Command	Query	Options and Query Returns
:SEARch:TRANSition:QUALifier <qualifier> (see <a href="#">page 946</a> )	:SEARch:TRANSition:QUALifier? (see <a href="#">page 946</a> )	<qualifier> ::= {GREaterthan   LESSthan}
:SEARch:TRANSition:SLOPe <slope> (see <a href="#">page 947</a> )	:SEARch:TRANSition:SLOPe? (see <a href="#">page 947</a> )	<slope> ::= {NEGative   POSitive}
:SEARch:TRANSition:SOURce <source> (see <a href="#">page 948</a> )	:SEARch:TRANSition:SOURce? (see <a href="#">page 948</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARch:TRANSition:TIME <time>[suffix] (see <a href="#">page 949</a> )	:SEARch:TRANSition:TIME? (see <a href="#">page 949</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**:SEARch:TRANSition:QUALifier****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:TRANSition:QUALifier <qualifier>`  
`<qualifier> ::= {GREaterthan | LESSthan}`

The :SEARch:TRANSition:QUALifier command specifies whether to search for edge transitions greater than or less than a time.

**Query Syntax**    `:SEARch:TRANSition:QUALifier?`

The :SEARch:TRANSition:QUALifier? query returns the current transition search qualifier.

**Return Format**    `<qualifier><NL>`  
`<qualifier> ::= {GRE | LESS}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:MODE"](#) on page 923
- [":SEARch:TRANSition:TIME"](#) on page 949

**:SEARch:TRANSition:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:TRANSition:SLOPe <slope>`  
`<slope> ::= {NEGative | POSitive}`

The :SEARch:TRANSition:SLOPe command selects whether to search for rising edge (POSitive slope) transitions or falling edge (NEGative slope) transitions.

**Query Syntax**    `:SEARch:TRANSition:SLOPe?`

The :SEARch:TRANSition:SLOPe? query returns the current transition search slope setting.

**Return Format**    `<slope><NL>`  
`<slope> ::= {NEG | POS}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [“:SEARch:MODE”](#) on page 923
- [“:SEARch:TRANSition:SOURce”](#) on page 948
- [“:SEARch:TRANSition:TIME”](#) on page 949

## :SEARch:TRANSition:SOURce

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:TRANSition:SOURce <source>`

`<source> ::= CHANnel<n>`

`<n> ::= 1 to (# analog channels) in NR1 format`

The :SEARch:TRANSition:SOURce command selects the channel on which to search for edge transitions.

**Query Syntax**    `:SEARch:TRANSition:SOURce?`

The :SEARch:TRANSition:SOURce? query returns the current transition search source.

**Return Format**    `<source><NL>`

`<source> ::= CHAN<n>`

- See Also**
- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
  - [":SEARch:MODE" on page 923](#)
  - [":SEARch:TRANSition:SLOPe" on page 947](#)

## :SEARch:TRANSition:TIME

**N** (see [page 1334](#))

**Command Syntax** :SEARch:TRANSition:TIME <time>[suffix]

<time> ::= floating-point number in NR3 format

[suffix] ::= {s | ms | us | ns | ps}

The :SEARch:TRANSition:TIME command sets the time of the transition to search for. You can search for transitions greater than or less than this time.

**Query Syntax** :SEARch:TRANSition:TIME?

The :SEARch:TRANSition:TIME? query returns the current transition time value.

**Return Format** <time><NL>

<time> ::= floating-point number in NR3 format

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [“:SEARch:TRANSition:QUALifier”](#) on page 946

## :SEARch:SERial:A429 Commands

**Table 127** :SEARch:SERial:A429 Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:A429:L ABel <value> (see <a href="#">page 951</a> )	:SEARch:SERial:A429:L ABel? (see <a href="#">page 951</a> )	<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string> from 0-255 <hex> ::= #Hnn where n ::= {0,...,9   A,...,F} <octal> ::= #Qnnn where n ::= {0,...,7} <string> ::= "0xnn" where n ::= {0,...,9   A,...,F}
:SEARch:SERial:A429:M ODE <condition> (see <a href="#">page 952</a> )	:SEARch:SERial:A429:M ODE? (see <a href="#">page 952</a> )	<condition> ::= {LABEL   LBITS   PERRor   WERRor   GERRor   WGERRors   ALLerrors}
:SEARch:SERial:A429:P ATTern:DATA <string> (see <a href="#">page 953</a> )	:SEARch:SERial:A429:P ATTern:DATA? (see <a href="#">page 953</a> )	<string> ::= "nn...n" where n ::= {0   1}, length depends on FORMat
:SEARch:SERial:A429:P ATTern:SDI <string> (see <a href="#">page 954</a> )	:SEARch:SERial:A429:P ATTern:SDI? (see <a href="#">page 954</a> )	<string> ::= "nn" where n ::= {0   1}, length always 2 bits
:SEARch:SERial:A429:P ATTern:SSM <string> (see <a href="#">page 955</a> )	:SEARch:SERial:A429:P ATTern:SSM? (see <a href="#">page 955</a> )	<string> ::= "nn" where n ::= {0   1}, length always 2 bits

## :SEARch:SERial:A429:LABel

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:A429:LABel <value>`

`<value> ::= 8-bit integer in decimal, <hex>, <octal>, or <string>`  
                   `from 0-255`

`<hex> ::= #Hnn where n ::= {0,...,9 | A,...,F}`

`<octal> ::= #Qnnn where n ::= {0,...,7}`

`<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}`

The :SEARch:SERial:A429:LABel command defines the ARINC 429 label value when labels are used in the selected search mode.

**Query Syntax**    `:SEARch:SERial:A429:LABel?`

The :SEARch:SERial:A429:LABel? query returns the current label value in decimal format.

**Return Format**    `<value><NL>` in decimal format

**Errors**

- ["-241, Hardware missing" on page 1293](#)

**See Also**

- ["Introduction to :TRIGger Commands" on page 1047](#)
- [":SEARch:SERial:A429:MODE" on page 952](#)

## :SEARch:SERial:A429:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:A429:MODE <condition>`

`<condition> ::= {LABel | LBITS | PERRor | WERRor | GERRor | WGERRors | ALLerrors}`

The :SEARch:SERial:A429:MODE command selects the type of ARINC 429 information to find in the Lister display:

- LABel – finds the specified label value.
- LBITS – finds the label and the other word fields as specified.
- PERRor – finds words with a parity error.
- WERRor – finds an intra-word coding error.
- GERRor – finds an inter-word gap error.
- WGERRors – finds either a Word or Gap Error.
- ALLerrors – finds any of the above errors.

**Query Syntax**    `:SEARch:SERial:A429:MODE?`

The :SEARch:SERial:A429:MODE? query returns the current ARINC 429 search mode condition.

**Return Format**    `<condition><NL>`

`<condition> ::= {LAB | LBIT | PERR | WERR | GERR | WGER | ALL}`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :SBUS<n> Commands" on page 723](#)  
                 • [":SBUS<n>:MODE" on page 727](#)  
                 • [":SEARch:SERial:A429:LABel" on page 951](#)  
                 • [":SEARch:SERial:A429:PATTERn:DATA" on page 953](#)  
                 • [":SEARch:SERial:A429:PATTERn:SDI" on page 954](#)  
                 • [":SEARch:SERial:A429:PATTERn:SSM" on page 955](#)  
                 • [":SBUS<n>:A429:SOURce" on page 737](#)

## :SEARch:SERial:A429:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:A429:PATTERn:DATA <string>

<string> ::= "nn...n" where n ::= {0 | 1}, length depends on FORMAT

The :SEARch:SERial:A429:PATTERn:DATA command defines the ARINC 429 data pattern resource according to the string parameter. This pattern controls the data pattern searched for in each ARINC 429 word.

### NOTE

If more bits are sent for <string> than specified by the :SBUS<n>:A429:FORMAT command, the most significant bits will be truncated.

**Query Syntax** :SEARch:SERial:A429:PATTERn:DATA?

The :SEARch:SERial:A429:PATTERn:DATA? query returns the current settings of the specified ARINC 429 data pattern resource in the binary string format.

**Return Format** <string><NL>

**Errors**

- ["-241, Hardware missing" on page 1293](#)

**See Also**

- ["Introduction to :TRIGger Commands" on page 1047](#)
- [":SEARch:SERial:A429:MODE" on page 952](#)
- [":SEARch:SERial:A429:PATTERn:SDI" on page 954](#)
- [":SEARch:SERial:A429:PATTERn:SSM" on page 955](#)

**:SEARch:SERial:A429:PATTERn:SDI****N** (see [page 1334](#))**Command Syntax**    `:SEARch:SERial:A429:PATTERn:SDI <string>``<string> ::= "nn" where n ::= {0 | 1}, length always 2 bits`

The :SEARch:SERial:A429:PATTERn:SDI command defines the ARINC 429 two-bit SDI pattern resource according to the string parameter. This pattern controls the SDI pattern searched for in each ARINC 429 word.

The specified SDI is only used if the :SBUS<n>:A429:FORMAT includes the SDI field.

**Query Syntax**    `:SEARch:SERial:A429:PATTERn:SDI?`

The :SEARch:SERial:A429:PATTERn:SDI? query returns the current settings of the specified ARINC 429 two-bit SDI pattern resource in the binary string format.

**Return Format**    `<string><NL>`**Errors**

- 241, [Hardware missing](#) on page 1293

**See Also**

- [Introduction to :TRIGger Commands](#) on page 1047

- [:SBUS<n>:A429:FORMAT](#) on page 735

- [:SEARch:SERial:A429:MODE](#) on page 952

- [:SEARch:SERial:A429:PATTERn:DATA](#) on page 953

- [:SEARch:SERial:A429:PATTERn:SSM](#) on page 955

## :SEARch:SERial:A429:PATTERn:SSM

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:A429:PATTERn:SSM <string>`

```
<string> ::= "nn" where n ::= {0 | 1}, length always 2 bits
```

The :SEARch:SERial:A429:PATTERn:SSM command defines the ARINC 429 two-bit SSM pattern resource according to the string parameter. This pattern controls the SSM pattern searched for in each ARINC 429 word.

The specified SSM is only used if the :SBUS<n>:A429:FORMat includes the SSM field.

**Query Syntax**    `:SEARch:SERial:A429:PATTERn:SSM?`

The :SEARch:SERial:A429:PATTERn:SSM? query returns the current settings of the specified ARINC 429 two-bit SSM pattern resource in the binary string format.

**Return Format**    `<string><NL>`

**Errors**    • ["-241, Hardware missing" on page 1293](#)

**See Also**    • ["Introduction to :TRIGger Commands" on page 1047](#)

- [":SBUS<n>:A429:FORMat" on page 735](#)

- [":SEARch:SERial:A429:MODE" on page 952](#)

- [":SEARch:SERial:A429:PATTERn:DATA" on page 953](#)

- [":SEARch:SERial:A429:PATTERn:SDI" on page 954](#)

## :SEARch:SERial:CAN Commands

**Table 128**:SEARch:SERial:CAN Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:CAN:MODE <value> (see page 957)	:SEARch:SERial:CAN:MODE? (see page 957)	<value> ::= { IDData   DATA   IDRremote   IDEither   ERRor   ACKerror   FORMerror   STUFFerror   CRCerror   ALLerrors   OVERload   MESSAGE   MSIGnal}
:SEARch:SERial:CAN:ATTern:DATA <string> (see page 959)	:SEARch:SERial:CAN:ATTern:DATA? (see page 959)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} for hexadecimal
:SEARch:SERial:CAN:ATTern:DATA:LENGTH <length> (see page 960)	:SEARch:SERial:CAN:ATTern:DATA:LENGTH? (see page 960)	<length> ::= integer from 1 to 8 in NR1 format
:SEARch:SERial:CAN:ATTern:ID <string> (see page 961)	:SEARch:SERial:CAN:ATTern:ID? (see page 961)	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} for hexadecimal
:SEARch:SERial:CAN:ATTern:ID:MODE <value> (see page 962)	:SEARch:SERial:CAN:ATTern:ID:MODE? (see page 962)	<value> ::= {STANDARD   EXTENDED}
:SEARch:SERial:CAN:SYMBolic:MESSAge <name> (see page 963)	:SEARch:SERial:CAN:SYMBolic:MESSAge? (see page 963)	<name> ::= quoted ASCII string
:SEARch:SERial:CAN:SYMBolic:SIGNAl <name> (see page 964)	:SEARch:SERial:CAN:SYMBolic:SIGNAl? (see page 964)	<name> ::= quoted ASCII string
:SEARch:SERial:CAN:SYMBolic:VALue <data> (see page 965)	:SEARch:SERial:CAN:SYMBolic:VALue? (see page 965)	<data> ::= value in NR3 format

## :SEARch:SERial:CAN:MODE

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:CAN:MODE <value>

```
<value> ::= { IDData | DATA | IDR | IDE | ERR | ACK | FORM | STUF | CRC
    | ALL | OVER | MESS | MSIG }
```

The :SEARch:SERial:CAN:MODE command selects the type of CAN information to find in the Lister display:

Condition	Front-panel name	Description
IDData	Data Frame ID	Finds data frames matching the specified ID.
DATA	Data Frame ID and Data	Finds data frames matching the specified ID and data.
IDRemote	Remote Frame ID	Finds remote frames with the specified ID.
IDEither	Remote or Data Frame ID	Finds remote or data frames matching the specified ID.
ERRor	Error Frame	Finds CAN active error frames.
ACKerror	Acknowledge Error	Finds the acknowledge bit if the polarity is incorrect.
FORMrror	Form Error	Finds reserved bit errors.
STUFFerror	Stuff Error	Finds 6 consecutive 1s or 6 consecutive 0s, while in a non-error or non overload frame.
CRCerror	CRC Error	Finds when the calculated CRC does not match the transmitted CRC.
ALLerrors	All Errors	Finds any form error or active error.
OVERload	Overload Frame	Finds CAN overload frames.
MESSage	Message	Finds a symbolic message.
MSIGnal	Message and Signal (non-FD)	Finds a symbolic message and a signal value.

**Query Syntax** :SEARch:SERial:CAN:MODE?

The :SEARch:SERial:CAN:MODE? query returns the currently selected mode.

**Return Format** <value><NL>

```
<value> ::= { IDD | DATA | IDR | IDE | ERR | ACK | FORM | STUF | CRC
    | ALL | OVER | MESS | MSIG }
```

- See Also**
- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
  - [“:SEARch:SERial:CAN:PATTERn:DATA” on page 959](#)
  - [“:SEARch:SERial:CAN:PATTERn:ID” on page 961](#)

- "[:RECall:DBC\[:STARt\]](#)" on page 686
- "[:SEARch:SERial:CAN:SYMBolic:MESSAge](#)" on page 963
- "[:SEARch:SERial:CAN:SYMBolic:SIGNAl](#)" on page 964
- "[:SEARch:SERial:CAN:SYMBolic:VALUe](#)" on page 965

## :SEARch:SERial:CAN:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:CAN:PATTERn:DATA <string>

```
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
for hexadecimal
```

The :SEARch:SERial:CAN:PATTERn:DATA command specifies the data value when searching for Data Frame ID and Data.

The length of the data value is specified using the :SEARch:SERial:CAN:PATTERn:DATA:LENGth command.

**Query Syntax** :SEARch:SERial:CAN:PATTERn:DATA?

The :SEARch:SERial:CAN:PATTERn:DATA? query returns the current data value setting.

**Return Format** <string><NL>

```
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
for hexadecimal
```

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:CAN:MODE"](#) on page 957
- [":SEARch:SERial:CAN:PATTERn:DATA:LENGth"](#) on page 960

**:SEARch:SERial:CAN:PATTERn:DATA:LENGth**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:CAN:PATTERn:DATA:LENGth <length>`

`<length> ::= integer from 1 to 8 in NR1 format`

The :SEARch:SERial:CAN:PATTERn:DATA:LENGth command specifies the length of the data value when searching for Data Frame ID and Data.

The data value is specified using the :SEARch:SERial:CAN:PATTERn:DATA command.

**Query Syntax**    `:SEARch:SERial:CAN:PATTERn:DATA:LENGth?`

The :SEARch:SERial:CAN:PATTERn:DATA:LENGth? query returns the current data length setting.

**Return Format**    `<length><NL>`

`<length> ::= integer from 1 to 8 in NR1 format`

**See Also**

- [Chapter 30, “:SEARch Commands,” starting on page 919](#)

- [“:SEARch:SERial:CAN:MODE” on page 957](#)

- [“:SEARch:SERial:CAN:PATTERn:DATA” on page 959](#)

## :SEARch:SERial:CAN:PATTERn:ID

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:CAN:PATTERn:ID <string>

```
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
           for hexadecimal
```

The :SEARch:SERial:CAN:PATTERn:ID command specifies the ID value when searching for a CAN event.

The value can be a standard ID or an extended ID, depending on the :SEARch:SERial:CAN:PATTERn:ID:MODE command's setting.

**Query Syntax** :SEARch:SERial:CAN:PATTERn:ID?

The :SEARch:SERial:CAN:PATTERn:ID? query returns the current ID value setting.

**Return Format** <string><NL>

```
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
           for hexadecimal
```

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:CAN:MODE"](#) on page 957
- [":SEARch:SERial:CAN:PATTERn:ID:MODE"](#) on page 962

**:SEARch:SERial:CAN:PATTERn:ID:MODE****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:CAN:PATTERn:ID:MODE <value>`  
`<value> ::= {STANDARD | EXTended}`

The :SEARch:SERial:CAN:PATTERn:ID:MODE command specifies whether a standard ID value or an extended ID value is used when searching for a CAN event.

The ID value is specified using the :SEARch:SERial:CAN:PATTERn:ID command.

**Query Syntax**    `:SEARch:SERial:CAN:PATTERn:ID:MODE?`

The :SEARch:SERial:CAN:PATTERn:ID:MODE? query returns the current setting.

**Return Format**    `<value><NL>`  
`<value> ::= {STAN | EXT}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:CAN:MODE"](#) on page 957
- [":SEARch:SERial:CAN:PATTERn:ID"](#) on page 961

## :SEARch:SERial:CAN:SYMBolic:MESSAge

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:CAN:SYMBolic:MESSAge <name>`  
`<name> ::= quoted ASCII string`

The :SEARch:SERial:CAN:SYMBolic:MESSAge command specifies the message to search for when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN serial search mode is set to MESSage or MSIGnal.

**Query Syntax**    `:SEARch:SERial:CAN:SYMBolic:MESSAge?`

The :SEARch:SERial:CAN:SYMBolic:MESSAge? query returns the specified message.

**Return Format**    `<name><NL>`  
`<name> ::= quotes ASCII string`

**See Also**

- "[:RECall:DBC\[:STARt\]](#)" on page 686
- "[:SEARch:SERial:CAN:MODE](#)" on page 957
- "[:SEARch:SERial:CAN:SYMBolic:SIGNAl](#)" on page 964
- "[:SEARch:SERial:CAN:SYMBolic:VALUe](#)" on page 965

**:SEARch:SERial:CAN:SYMBolic:SIGNAl****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:CAN:SYMBolic:SIGNAl <name>`  
                  `<name> ::= quoted ASCII string`

The :SEARch:SERial:CAN:SYMBolic:SIGNAl command specifies the signal to search for when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN serial search mode is set to MSIGnAl.

**Query Syntax**    `:SEARch:SERial:CAN:SYMBolic:SIGNAl?`  
The :SEARch:SERial:CAN:SYMBolic:SIGNAl? query returns the specified signal.

**Return Format**    `<name><NL>`  
                  `<name> ::= quoted ASCII string`

**See Also**

- [":RECall:DBC\[:START\]" on page 686](#)
- [":SEARch:SERial:CAN:MODE" on page 957](#)
- [":SEARch:SERial:CAN:SYMBolic:MESSAge" on page 963](#)
- [":SEARch:SERial:CAN:SYMBolic:VALUe" on page 965](#)

## :SEARch:SERial:CAN:SYMBolic:VALue

**N** (see [page 1334](#))

**Command Syntax**

```
:SEARch:SERial:CAN:SYMBolic:VALue <data>
<data> ::= value in NR3 format
```

The :SEARch:SERial:CAN:SYMBolic:VALue command specifies the signal value to search for when CAN symbolic data has been loaded (recalled) into the oscilloscope and the CAN serial search mode is set to MSIGnal.

**NOTE** Encoded signal values are not supported in the remote interface (even though they can be used in the front panel graphical interface).

---

**Query Syntax**

```
:SEARch:SERial:CAN:SYMBolic:VALue?
```

The :SEARch:SERial:CAN:SYMBolic:VALue? query returns the specified signal value.

**Return Format**

```
<data><NL>
<data> ::= value in NR3 format
```

- See Also**
- "[:RECall:DBC\[:STARt\]](#)" on page 686
  - "[:SEARch:SERial:CAN:MODE](#)" on page 957
  - "[:SEARch:SERial:CAN:SYMBolic:MESSage](#)" on page 963
  - "[:SEARch:SERial:CAN:SYMBolic:SIGNal](#)" on page 964

## :SEARch:SERial:FLEXray Commands

**Table 129** :SEARch:SERial:FLEXray Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:FLEXra y:CYCLE <cycle> (see <a href="#">page 967</a> )	:SEARch:SERial:FLEXra y:CYCLE? (see <a href="#">page 967</a> )	<cycle> ::= {ALL   <cycle #>} <cycle #> ::= integer from 0-63
:SEARch:SERial:FLEXra y:DATA <string> (see <a href="#">page 968</a> )	:SEARch:SERial:FLEXra y:DATA? (see <a href="#">page 968</a> )	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X }
:SEARch:SERial:FLEXra y:DATA:LENGth <length> (see <a href="#">page 969</a> )	:SEARch:SERial:FLEXra y:DATA:LENGth? (see <a href="#">page 969</a> )	<length> ::= integer from 1 to 12 in NR1 format
:SEARch:SERial:FLEXra y:FRAMe <frame id> (see <a href="#">page 970</a> )	:SEARch:SERial:FLEXra y:FRAMe? (see <a href="#">page 970</a> )	<frame_id> ::= {ALL   <frame #>} <frame #> ::= integer from 1-2047
:SEARch:SERial:FLEXra y:MODE <value> (see <a href="#">page 971</a> )	:SEARch:SERial:FLEXra y:MODE? (see <a href="#">page 971</a> )	<value> ::= {FRAMe   CYCLE   DATA   HERRor   FERRor   AERRor}

**:SEARch:SERial:FLEXray:CYCLe****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:FLEXray:CYCLe <cycle>`  
`<cycle> ::= {ALL | <cycle #>}`  
`<cycle #> ::= integer from 0-63`

The :SEARch:SERial:FLEXray:CYCLe command specifies the cycle value to find when searching for FlexRay frames.

A cycle value of -1 is the same as ALL.

**Query Syntax**    `:SEARch:SERial:FLEXray:CYCLe?`

The :SEARch:SERial:FLEXray:CYCLe? query returns the current cycle value setting.

**Return Format**    `<cycle><NL>`  
`<cycle> ::= {ALL | <cycle #>}`  
`<cycle #> ::= integer from 0-63`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:FLEXray:MODE"](#) on page 971

**:SEARch:SERial:FLEXray:DATA****N** (see [page 1334](#))**Command Syntax**    `:SEARch:SERial:FLEXray:DATA <string>``<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}`

The :SEARch:SERial:FLEXray:DATA command specifies the data value to find when searching for FlexRay frames.

The length of the data value is specified by the :SEARch:SERial:FLEXray:DATA:LENGth command.

**Query Syntax**    `:SEARch:SERial:FLEXray:DATA?`

The :SEARch:SERial:FLEXray:DATA? query returns the current data value setting.

**Return Format**    `<string><NL>``<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}`**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:FLEXray:MODE"](#) on page 971
- [":SEARch:SERial:FLEXray:DATA:LENGth"](#) on page 969

**:SEARch:SERial:FLEXray:DATA:LENGth**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:FLEXray:DATA:LENGth <length>`  
`<length> ::= integer from 1 to 12 in NR1 format`

The :SEARch:SERial:FLEXray:DATA:LENGth command specifies the length of data values when searching for FlexRay frames.

The data value is specified using the :SEARch:SERial:FLEXray:DATA command.

**Query Syntax**    `:SEARch:SERial:FLEXray:DATA:LENGth?`

The :SEARch:SERial:FLEXray:DATA:LENGth? query returns the current data length setting.

**Return Format**    `<length><NL>`  
`<length> ::= integer from 1 to 12 in NR1 format`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:FLEXray:MODE"](#) on page 971
- [":SEARch:SERial:FLEXray:DATA"](#) on page 968

## :SEARch:SERial:FLEXray:FRAMe

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:FLEXray:FRAMe <frame_id>`  
`<frame_id> ::= {ALL | <frame #>}`  
`<frame #> ::= integer from 1-2047`

The :SEARch:SERial:FLEXray:FRAMe command specifies the frame ID value to find when searching for FlexRay frames.

**Query Syntax**    `:SEARch:SERial:FLEXray:FRAMe?`

The :SEARch:SERial:FLEXray:FRAMe? query returns the current frame ID setting.

**Return Format**    `<frame_id><NL>`  
`<frame_id> ::= {ALL | <frame #>}`  
`<frame #> ::= integer from 1-2047`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:FLEXray:MODE"](#) on page 971

## :SEARch:SERial:FLEXray:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:FLEXray:MODE <value>`

`<value> := {FRAMe | CYCLe | DATA | HERRor | FERRor | AERRor}`

The :SEARch:SERial:FLEXray:MODE command selects the type of FlexRay information to find in the Lister display:

- FRAMe – searches for FlexRay frames with the specified frame ID.
- CYCLe – searches for FlexRay frames with the specified cycle number and frame ID.
- DATA – searches for FlexRay frames with the specified data, cycle number, and frame ID.
- HERRor – searches for header CRC errors.
- FERRor – searches for frame CRC errors.
- AERRor – searches for all errors.

**Query Syntax**    `:SEARch:SERial:FLEXray:MODE?`

The :SEARch:SERial:FLEXray:MODE? query returns the currently selected mode.

**Return Format**    `<value><NL>`

`<value> := {FRAM | CYCL | DATA | HERR | FERR | AERR}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:FLEXray:FRAMe"](#) on page 970
- [":SEARch:SERial:FLEXray:CYCLE"](#) on page 967
- [":SEARch:SERial:FLEXray:DATA"](#) on page 968

## :SEARch:SERial:I2S Commands

**Table 130** :SEARch:SERial:I2S Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:I2S:AU Dio <audio_ch> (see page 973)	:SEARch:SERial:I2S:AU Dio? (see page 973)	<audio_ch> ::= {RIGHT   LEFT   EITHer}
:SEARch:SERial:I2S:MO DE <value> (see page 974)	:SEARch:SERial:I2S:MO DE? (see page 974)	<value> ::= {EQUAL   NOTequal   LESSthan   GREaterthan   INRange   OUTRange}
:SEARch:SERial:I2S:PA TTern:DATA <string> (see page 975)	:SEARch:SERial:I2S:PA TTern:DATA? (see page 975)	<string> ::= "n" where n ::= 32-bit integer in signed decimal when <base> = DECimal <string> ::= "nn...n" where n ::= {0   1   X} when <base> = BINary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X} when <base> = HEX
:SEARch:SERial:I2S:PA TTern:FORMAT <base> (see page 976)	:SEARch:SERial:I2S:PA TTern:FORMAT? (see page 976)	<base> ::= {BINary   HEX   DECimal}
:SEARch:SERial:I2S:RA NGe <lower>, <upper> (see page 977)	:SEARch:SERial:I2S:RA NGe? (see page 977)	<lower> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <upper> ::= 32-bit integer in signed decimal, <nondecimal>, or <string> <nondecimal> ::= #Hnn...n where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F} for hexadecimal

**:SEARch:SERial:I2S:AUDIO**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:I2S:AUDIo <audio_ch>`  
`<audio_ch> ::= {RIGHT | LEFT | EITHer}`

The :SEARch:SERial:I2S:AUDIO command specifies the channel on which to search for I2S events: right, left, or either channel.

**Query Syntax**    `:SEARch:SERial:I2S:AUDIo?`

The :SEARch:SERial:I2S:AUDIO? query returns the current channel setting.

**Return Format**    `<audio_ch><NL>`  
`<audio_ch> ::= {RIGH | LEFT | EITH}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [“:SEARch:SERial:I2S:MODE”](#) on page 974

## :SEARch:SERial:I2S:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:I2S:MODE <value>`

```
<value> ::= {EQUAL | NOTequal | LESSthan | GREaterthan | INRange
              | OUTRange}
```

The :SEARch:SERial:I2S:MODE command selects the type of I2S information to find in the Lister display:

- EQUAL – searches for the specified audio channel's data word when it equals the specified word.
- NOTequal – searches for any word other than the specified word.
- LESSthan – searches for channel data words less than the specified value.
- GREaterthan – searches for channel data words greater than the specified value.
- INRange – searches for channel data words in the range.
- OUTRange – searches for channel data words outside the range.

Data word values are specified using the :SEARch:SERial:I2S:PATTERn:DATA command.

Value ranges are specified using the :SEARch:SERial:I2S:RANGe command.

**Query Syntax**    `:SEARch:SERial:I2S:MODE?`

The :SEARch:SERial:I2S:MODE? query returns the currently selected mode.

**Return Format**    `<value><NL>`

```
<value> ::= {EQU | NOT | LESS | GRE | INR | OUTR}
```

- See Also**
- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
  - [":SEARch:SERial:I2S:PATTERn:DATA" on page 975](#)
  - [":SEARch:SERial:I2S:RANGe" on page 977](#)
  - [":SEARch:SERial:I2S:AUDIO" on page 973](#)

## :SEARch:SERial:I2S:PATTERn:DATA

**N** (see [page 1334](#))

**Command Syntax**

```
:SEARch:SERial:I2S:PATTERn:DATA <string>
<string> ::= "n" where n ::= 32-bit integer in signed decimal
           when <base> = DECimal
<string> ::= "nn...n" where n ::= {0 | 1 | X} when <base> = BINary
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
           when <base> = HEX
```

The :SEARch:SERial:I2S:PATTERn:DATA command specifies the data word value when searching for I2S events.

The base of the value entered with this command is specified using the :SEARch:SERial:I2S:PATTERn:FORMAT command.

**Query Syntax**

```
:SEARch:SERial:I2S:PATTERn:DATA?
```

The :SEARch:SERial:I2S:PATTERn:DATA? query returns the current data word value setting.

**Return Format**

```
<string><NL>
<string> ::= "n" where n ::= 32-bit integer in signed decimal
           when <base> = DECimal
<string> ::= "nn...n" where n ::= {0 | 1 | X} when <base> = BINary
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}
           when <base> = HEX
```

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:I2S:MODE"](#) on page 974
- [":SEARch:SERial:I2S:PATTERn:FORMAT"](#) on page 976

**:SEARch:SERial:I2S:PATTERn:FORMAT**(see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:I2S:PATTERn:FORMAT <base>`  
`<base> ::= {BINary | HEX | DECimal}`

The :SEARch:SERial:I2S:PATTERn:FORMAT command specifies the number base used with the :SEARch:SERial:I2S:PATTERn:DATA command.

**Query Syntax**    `:SEARch:SERial:I2S:PATTERn:FORMAT?`

The :SEARch:SERial:I2S:PATTERn:FORMAT? query returns the current number base setting.

**Return Format**    `<base><NL>`  
`<base> ::= {BIN | HEX | DEC}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:I2S:PATTERn:DATA"](#) on page 975

## :SEARch:SERial:I2S:RANGE



(see [page 1334](#))

### Command Syntax

```
:SEARch:SERial:I2S:RANGE <lower>, <upper>
<lower> ::= 32-bit integer in signed decimal, <nondecimal>, or <string>
<upper> ::= 32-bit integer in signed decimal, <nondecimal>, or <string>
<nondecimal> ::= #Hnn...n where n ::= {0,...,9 | A,...,F} for hexadecimal
<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal
```

The :SEARch:SERial:I2S:RANGE command specifies the data value range when searching for I2S events in the INRange and OUTRange search modes (set by the :SEARch:SERial:I2S:MODE command).

You can enter the parameters in any order – the smaller value becomes the <lower> and the larger value becomes the <upper>.

### Query Syntax

```
:SEARch:SERial:I2S:RANGE?
```

The :SEARch:SERial:I2S:RANGE? query returns the current data value range setting.

### Return Format

```
<lower>, <upper><NL>
<lower> ::= 32-bit integer in signed decimal
<upper> ::= 32-bit integer in signed decimal
```

### See Also

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:I2S:MODE"](#) on page 974

## :SEARch:SERial:IIC Commands

**Table 131** :SEARch:SERial:IIC Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:IIC:MO DE <value> (see <a href="#">page 979</a> )	:SEARch:SERial:IIC:MO DE? (see <a href="#">page 979</a> )	<value> ::= { READ7   WRITE7   NACKnowledge   ANACK   R7Data2   W7Data2   RESTart   READEprom}
:SEARch:SERial:IIC:PA TTern:ADDRess <value> (see <a href="#">page 981</a> )	:SEARch:SERial:IIC:PA TTern:ADDRess? (see <a href="#">page 981</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:PA TTern:DATA <value> (see <a href="#">page 982</a> )	:SEARch:SERial:IIC:PA TTern:DATA? (see <a href="#">page 982</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:PA TTern:DATA2 <value> (see <a href="#">page 983</a> )	:SEARch:SERial:IIC:PA TTern:DATA2? (see <a href="#">page 983</a> )	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,...,9   A,...,F}
:SEARch:SERial:IIC:QU ALifier <value> (see <a href="#">page 984</a> )	:SEARch:SERial:IIC:QU ALifier? (see <a href="#">page 984</a> )	<value> ::= { EQUAL   NOTequal   LESSthan   GREaterthan}

## :SEARch:SERial:IIC:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:IIC:MODE <value>`

```
<value> ::= {READ7 | WRITE7 | NACKnowledge | ANACK | R7Data2
             | W7Data2 | RESTart | READEprom}
```

The :SEARch:SERial:IIC:MODE command selects the type of IIC information to find in the Lister display:

- READ7 – searches for 7-bit address frames containing Start:Address7:Read:Ack:Data. The value READ is also accepted for READ7.
- WRITE7 – searches for 7-bit address frames containing Start:Address7:Write:Ack:Data. The value WRITE is also accepted for WRITE7.
- NACKnowledge – searches for missing acknowledge events.
- ANACK – searches for address with no acknowledge events.
- R7Data2 – searches for 7-bit address frames containing Start:Address7:Read:Ack:Data:Ack:Data2.
- W7Data2 – searches for 7-bit address frames containing Start:Address7:Write:Ack:Data:Ack:Data2.
- RESTart – searches for another start condition occurring before a stop condition.
- READEprom – searches for EEPROM data reads.

### NOTE

The short form of READ7 (READ7), READEprom (READE), and WRITE7 (WRIT7) do not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)).

When searching for events containing addresses, address values are specified using the :SEARch:SERial:IIC:PATTERn:ADDReSS command.

When searching for events containing data, data values are specified using the :SEARch:SERial:IIC:PATTERn:DATA and :SEARch:SERial:IIC:PATTERn:DATA2 commands.

**Query Syntax**    `:SEARch:SERial:IIC:MODE?`

The :SEARch:SERial:IIC:MODE? query returns the currently selected mode.

**Return Format**    `<value><NL>`

```
<value> ::= {READ7 | WRITE7 | NACK | ANAC | R7D2 | W7D2 | REST
             | READE}
```

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:SERial:IIC:PATTERn:ADDReSS"](#) on page 981
  - [":SEARch:SERial:IIC:PATTERn:DATA"](#) on page 982

- "[:SEARch:SERial:IIC:PATTERn:DATA2](#)" on page 983
- "[:SEARch:SERial:IIC:QUALifier](#)" on page 984

**:SEARch:SERial:IIC:PATTERn:ADDRess**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:IIC:PATTERn:ADDRess <value>`  
`<value> ::= integer or <string>`  
`<string> ::= "0xnn" n ::= {0,...,9 | A,...,F}`

The :SEARch:SERial:IIC:PATTERn:ADDRess command specifies address values when searching for IIC events.

To set don't care values, use the integer -1.

**Query Syntax**    `:SEARch:SERial:IIC:PATTERn:ADDRess?`

The :SEARch:SERial:IIC:PATTERn:ADDRess? query returns the current address value setting.

**Return Format**    `<value><NL>`  
`<value> ::= integer`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [“:SEARch:SERial:IIC:MODE”](#) on page 979

**:SEARch:SERial:IIC:PATTERn:DATA****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:IIC:PATTERn:DATA <value>`  
`<value> ::= integer or <string>`  
`<string> ::= "0xnn" n ::= {0,...,9 | A,...,F}`

The :SEARch:SERial:IIC:PATTERn:DATA command specifies data values when searching for IIC events.

To set don't care values, use the integer -1.

When searching for IIC EEPROM data read events, you specify the data value qualifier using the :SEARch:SERial:IIC:QUALifier command.

**Query Syntax**    `:SEARch:SERial:IIC:PATTERn:DATA?`

The :SEARch:SERial:IIC:PATTERn:DATA? query returns the current data value setting.

**Return Format**    `<value><NL>`

`<value> ::= integer`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:IIC:MODE"](#) on page 979
- [":SEARch:SERial:IIC:QUALifier"](#) on page 984
- [":SEARch:SERial:IIC:PATTERn:DATA2"](#) on page 983

## :SEARch:SERial:IIC:PATTERn:DATA2

**N** (see [page 1334](#))

**Command Syntax**

```
:SEARch:SERial:IIC:PATTERn:DATA2 <value>
<value> ::= integer or <string>
<string> ::= "0xnn" n ::= {0,...,9 | A,...,F}
```

The :SEARch:SERial:IIC:PATTERn:DATA2 command specifies the second data value when searching for IIC events with two data values.

To set don't care values, use the integer -1.

**Query Syntax**

```
:SEARch:SERial:IIC:PATTERn:DATA2?
```

The :SEARch:SERial:IIC:PATTERn:DATA2? query returns the current second data value setting.

**Return Format**

```
<value><NL>
<value> ::= integer
```

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:IIC:MODE"](#) on page 979
- [":SEARch:SERial:IIC:PATTERn:DATA"](#) on page 982

**:SEARch:SERial:IIC:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:SEARch:SERial:IIC:QUALifier <value>``<value> ::= {EQUAL | NOTEqual | LESSthan | GREaterthan}`

The :SEARch:SERial:IIC:QUALifier command specifies the data value qualifier used when searching for IIC EEPROM data read events.

**Query Syntax**    `:SEARch:SERial:IIC:QUALifier?`

The :SEARch:SERial:IIC:QUALifier? query returns the current data value qualifier setting.

**Return Format**    `<value><NL>``<value> ::= {EQU | NOT | LESS | GRE}`**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:IIC:MODE"](#) on page 979
- [":SEARch:SERial:IIC:PATTERn:DATA"](#) on page 982

## :SEARch:SERial:LIN Commands

**Table 132** :SEARch:SERial:LIN Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:LIN:ID <value> (see <a href="#">page 986</a> )	:SEARch:SERial:LIN:ID? (see <a href="#">page 986</a> )	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f (with Option AMS) <nondecimal> ::= #Hnn where n ::= {0,...,9   A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9   A,...,F} for hexadecimal
:SEARch:SERial:LIN:MO DE <value> (see <a href="#">page 987</a> )	:SEARch:SERial:LIN:MO DE? (see <a href="#">page 987</a> )	<value> ::= {ID   DATA   ERRor}
:SEARch:SERial:LIN:PA TTern:DATA <string> (see <a href="#">page 988</a> )	:SEARch:SERial:LIN:PA TTern:DATA? (see <a href="#">page 988</a> )	When :SEARch:SERial:LIN:PATTERn:FORMat DECimal, <string> ::= "n" where n ::= 32-bit integer in unsigned decimal, returns "\$" if data has any don't cares When :SEARch:SERial:LIN:PATTERn:FORMat HEX, <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X }
:SEARch:SERial:LIN:PA TTern:DATA:LENGTH <length> (see <a href="#">page 989</a> )	:SEARch:SERial:LIN:PA TTern:DATA:LENGTH? (see <a href="#">page 989</a> )	<length> ::= integer from 1 to 8 in NR1 format
:SEARch:SERial:LIN:PA TTern:FORMAT <base> (see <a href="#">page 990</a> )	:SEARch:SERial:LIN:PA TTern:FORMAT? (see <a href="#">page 990</a> )	<base> ::= {HEX   DECimal}
:SEARch:SERial:LIN:SY MBolic:FRAMe <name> (see <a href="#">page 991</a> )	:SEARch:SERial:LIN:SY MBolic:FRAMe? (see <a href="#">page 991</a> )	<name> ::= quoted ASCII string
:SEARch:SERial:LIN:SY MBolic:SIGNal <name> (see <a href="#">page 992</a> )	:SEARch:SERial:LIN:SY MBolic:SIGNal? (see <a href="#">page 992</a> )	<name> ::= quoted ASCII string
:SEARch:SERial:LIN:SY MBolic:VALue <data> (see <a href="#">page 993</a> )	:SEARch:SERial:LIN:SY MBolic:VALue? (see <a href="#">page 993</a> )	<data> ::= value in NR3 format

**:SEARch:SERial:LIN:ID****N** (see [page 1334](#))

- Command Syntax**    `:SEARch:SERial:LIN:ID <value>`
- `<value> ::= 7-bit integer in decimal, <nondecimal>, or <string>`  
`from 0-63 or 0x00-0x3f (with Option AMS)`
- `<nondecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal`
- `<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary`
- `<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F} for hexadecimal`
- The :SEARch:SERial:LIN:ID command specifies the frame ID value when searching for LIN events.
- Query Syntax**    `:SEARch:SERial:LIN:ID?`
- The :SEARch:SERial:LIN:ID? query returns the current frame ID setting.
- Return Format**    `<value><NL>`
- `<value> ::= 7-bit integer in decimal (with Option AMS)`
- See Also**
- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
  - [“:SEARch:SERial:LIN:MODE” on page 987](#)

## :SEARch:SERial:LIN:MODE

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:LIN:MODE <value>

<value> ::= { ID | DATA | ERRor | FRAMe | FSIGnal }

The :SEARch:SERial:LIN:MODE command selects the type of LIN information to find in the Lister display:

- ID – searches for a frame ID.
- DATA – searches for a frame ID and data.
- ERRor – searches for errors.
- FRAMe – searches for symbolic frames.
- FSIGnal – searched for symbolic frames and a signal values.

Frame IDs are specified using the :SEARch:SERial:LIN:ID command.

Data values are specified using the :SEARch:SERial:LIN:PATTern:DATA command.

Frames, signals, and signal values are specified using the :SEARch:SERial:LIN:SYMBolic:FRAMe, :SEARch:SERial:LIN:SYMBolic:SIGNal, and :SEARch:SERial:LIN:SYMBolic:VALue commands. LIN symbolic data files are loaded (recalled) using the :RECall:LDF[:START] command.

**Query Syntax** :SEARch:SERial:LIN:MODE?

The :SEARch:SERial:LIN:MODE? query returns the currently selected mode.

**Return Format** <value><NL>

<value> ::= { ID | DATA | ERR | FRAM | FSIG }

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:SERial:LIN:ID"](#) on page 986
  - [":SEARch:SERial:LIN:PATTern:DATA"](#) on page 988
  - [":RECall:LDF\[:START\]"](#) on page 688
  - [":SEARch:SERial:LIN:SYMBolic:FRAMe"](#) on page 991
  - [":SEARch:SERial:LIN:SYMBolic:SIGNal"](#) on page 992
  - [":SEARch:SERial:LIN:SYMBolic:VALue"](#) on page 993

## :SEARCh:SERial:LIN:PATTern:DATA

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SEARCh:SERial:LIN:PATTern:DATA &lt;string&gt;</code>
	When :SEARCh:SERial:LIN:PATTern:FORMAT DECimal, <code>&lt;string&gt; ::= "n"</code> where n ::= 32-bit integer in unsigned decimal
	When :SEARCh:SERial:LIN:PATTern:FORMAT HEX, <code>&lt;string&gt; ::= "0xnn...n"</code> where n ::= {0,...,9   A,...,F   X}
	The :SEARCh:SERial:LIN:PATTern:DATA command specifies the data value when searching for LIN events.
	The number base of the value entered with this command is specified using the :SEARCh:SERial:LIN:PATTern:FORMAT command. To set don't care values with the DATA command, the FORMAT must be HEX.
	The length of the data value entered is specified using the :SEARCh:SERial:LIN:PATTern:DATA:LENGth command.
<b>Query Syntax</b>	<code>:SEARCh:SERial:LIN:PATTern:DATA?</code>
	The :SEARCh:SERial:LIN:PATTern:DATA? query returns the current data value setting.
<b>Return Format</b>	<code>&lt;string&gt;&lt;NL&gt;</code>
	When :SEARCh:SERial:LIN:PATTern:FORMAT DECimal, <code>&lt;string&gt; ::= "n"</code> where n ::= 32-bit integer in unsigned decimal or "\$" if data has any don't cares
	When :SEARCh:SERial:LIN:PATTern:FORMAT HEX, <code>&lt;string&gt; ::= "0xnn...n"</code> where n ::= {0,...,9   A,...,F   X}
<b>See Also</b>	<ul style="list-style-type: none"> <li>• <a href="#">Chapter 30</a>, “:SEARCh Commands,” starting on page 919</li> <li>• <a href="#">":SEARCh:SERial:LIN:MODE"</a> on page 987</li> <li>• <a href="#">":SEARCh:SERial:LIN:PATTern:FORMAT"</a> on page 990</li> <li>• <a href="#">":SEARCh:SERial:LIN:PATTern:DATA:LENGth"</a> on page 989</li> </ul>

**:SEARch:SERial:LIN:PATTern:DATA:LENGth**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:LIN:PATTern:DATA:LENGth <length>`  
`<length> ::= integer from 1 to 8 in NR1 format`

The :SEARch:SERial:LIN:PATTern:DATA:LENGth command specifies the the length of the data value when searching for LIN events.

The data value is specified using the :SEARch:SERial:LIN:PATTern:DATA command.

**Query Syntax**    `:SEARch:SERial:LIN:PATTern:DATA:LENGth?`

The :SEARch:SERial:LIN:PATTern:DATA:LENGth? query returns the current data value length setting.

**Return Format**    `<length><NL>`  
`<length> ::= integer from 1 to 8 in NR1 format`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:LIN:PATTern:DATA"](#) on page 988

**:SEARch:SERial:LIN:PATTern:FORMAT**(see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:LIN:PATTern:FORMAT <base>`  
                  `<base> ::= {HEX | DECimal}`

The :SEARch:SERial:LIN:PATTern:FORMAT command specifies the number base used with the :SEARch:SERial:LIN:PATTern:DATA command.

**Query Syntax**    `:SEARch:SERial:LIN:PATTern:FORMAT?`

The :SEARch:SERial:LIN:PATTern:FORMAT? query returns the current number base setting.

**Return Format**    `<base><NL>`  
                  `<base> ::= {HEX | DEC}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:LIN:PATTern:DATA"](#) on page 988

**:SEARch:SERial:LIN:SYMBolic:FRAMe****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:LIN:SYMBolic:FRAMe <name>`  
                  `<name> ::= quoted ASCII string`

The :SEARch:SERial:LIN:SYMBolic:FRAMe command specifies the message to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FRAMe or FSIGnAl.

**Query Syntax**    `:SEARch:SERial:LIN:SYMBolic:FRAMe?`

The :SEARch:SERial:LIN:SYMBolic:FRAMe? query returns the specified message.

**Return Format**    `<name><NL>`  
                  `<name> ::= quotes ASCII string`

**See Also**

- [":RECall:LDF\[:STARt\]" on page 688](#)
- [":SEARch:SERial:LIN:MODE" on page 987](#)
- [":SEARch:SERial:LIN:SYMBolic:SIGNAl" on page 992](#)
- [":SEARch:SERial:LIN:SYMBolic:VALUe" on page 993](#)

**:SEARch:SERial:LIN:SYMBolic:SIGNAl****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:LIN:SYMBolic:SIGNAl <name>`  
                  `<name> ::= quoted ASCII string`

The :SEARch:SERial:LIN:SYMBolic:SIGNAl command specifies the signal to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FSIGnAl.

**Query Syntax**    `:SEARch:SERial:LIN:SYMBolic:SIGNAl?`

The :SEARch:SERial:LIN:SYMBolic:SIGNAl? query returns the specified signal.

**Return Format**    `<name><NL>`  
                  `<name> ::= quoted ASCII string`

**See Also**

- [":RECall:LDF\[:STARt\]" on page 688](#)
- [":SEARch:SERial:LIN:MODE" on page 987](#)
- [":SEARch:SERial:LIN:SYMBolic:FRAMe" on page 991](#)
- [":SEARch:SERial:LIN:SYMBolic:VALue" on page 993](#)

## :SEARch:SERial:LIN:SYMBolic:VALue

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:LIN:SYMBolic:VALue <data>`  
`<data> ::= value in NR3 format`

The :SEARch:SERial:LIN:SYMBolic:VALue command specifies the signal value to search for when LIN symbolic data has been loaded (recalled) into the oscilloscope and the LIN serial search mode is set to FSIGnal.

**NOTE** Encoded signal values are not supported in the remote interface (even though they can be used in the front panel graphical interface).

**Query Syntax**    `:SEARch:SERial:LIN:SYMBolic:VALue?`

The :SEARch:SERial:LIN:SYMBolic:VALue? query returns the specified signal value.

**Return Format**    `<data><NL>`  
`<data> ::= value in NR3 format`

**See Also**

- "[:RECall:LDF\[:START\]](#)" on page 688
- "[:SEARch:SERial:LIN:MODE](#)" on page 987
- "[:SEARch:SERial:LIN:SYMBolic:FRAMe](#)" on page 991
- "[:SEARch:SERial:LIN:SYMBolic:SIGNAl](#)" on page 992

## :SEARch:SERial:M1553 Commands

**Table 133** :SEARch:SERial:M1553 Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:M1553:MODE <value> (see <a href="#">page 995</a> )	:SEARch:SERial:M1553:MODE? (see <a href="#">page 995</a> )	<value> ::= {DSTARt   CSTArt   RTA   RTA11   PERRor   SERRor   MERRor}
:SEARch:SERial:M1553:PATTERn:DATA <string> (see <a href="#">page 996</a> )	:SEARch:SERial:M1553:PATTERn:DATA? (see <a href="#">page 996</a> )	<string> ::= "nn...n" where n ::= {0   1}
:SEARch:SERial:M1553:RTA <value> (see <a href="#">page 997</a> )	:SEARch:SERial:M1553:RTA? (see <a href="#">page 997</a> )	<value> ::= 5-bit integer in decimal, <hexadecimal>, <binary>, or <string> from 0-31 <hexadecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} <binary> ::= #Bnn...n where n ::= {0   1} for binary <string> ::= "0xnn" where n ::= {0,...,9 A,...,F}

## :SEARch:SERial:M1553:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:M1553:MODE <value>`  
`<value> ::= {DSTArt | CSTArt | RTA | RTA11 | PERRor | SERRor | MERRor}`

The :SEARch:SERial:M1553:MODE command selects the type of MIL-STD-1553 information to find in the Lister display:

- DSTArt – searches for the start of a Data word (at the end of a valid Data Sync pulse).
- CSTArt – searches for the start of a Command/Status word (at the end of a valid C/S Sync pulse).
- RTA – searches for the Remote Terminal Address (RTA) of a Command/Status word.
- RTA11 – searches for the Remote Terminal Address (RTA) and the additional 11 bits of a Command/Status word.
- PERRor – searches for (odd) parity errors for the data in the word.
- SERRor – searches for invalid Sync pulses.
- MERRor – searches for Manchester encoding errors.

In the RTA or RTA11 modes, the Remote Terminal Address is specified using the :SEARch:SERial:M1553:RTA command.

In the RTA11 mode, the additional 11 bits are specified using the :SEARch:SERial:M1553:PATTERn:DATA command.

**Query Syntax**    `:SEARch:SERial:M1553:MODE?`

The :SEARch:SERial:M1553:MODE? query returns the currently selected mode.

**Return Format**    `<value><NL>`  
`<value> ::= {DSTA | CSTA | RTA | RTA11 | PERR | SERR | MERR}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- “[:SEARch:SERial:M1553:RTA](#)” on page 997
- “[:SEARch:SERial:M1553:PATTERn:DATA](#)” on page 996

**:SEARch:SERial:M1553:PATTERn:DATA**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:M1553:PATTERn:DATA <string>`  
`<string> ::= "nn...n" where n ::= {0 | 1}`

The :SEARch:SERial:M1553:PATTERn:DATA command specifies the additional 11 bits when searching for the MIL-STD-1553 Remote Terminal Address + 11 Bits.

**Query Syntax**    `:SEARch:SERial:M1553:PATTERn:DATA?`

The :SEARch:SERial:M1553:PATTERn:DATA? query returns the current value setting for the additional 11 bits.

**Return Format**    `<string><NL>`  
`<string> ::= "nn...n" where n ::= {0 | 1}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:M1553:MODE"](#) on page 995

## :SEARch:SERial:M1553:RTA

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SEARch:SERial:M1553:RTA &lt;value&gt;</code>
	<code>&lt;value&gt;</code> ::= 5-bit integer in decimal, <code>&lt;hexadecimal&gt;</code> , <code>&lt;binary&gt;</code> , or <code>&lt;string&gt;</code> from 0-31
	<code>&lt;hexadecimal&gt;</code> ::= #Hnn where n ::= {0,...,9 A,...,F}
	<code>&lt;binary&gt;</code> ::= #Bnn...n where n ::= {0   1} for binary
	<code>&lt;string&gt;</code> ::= "0xnn" where n ::= {0,...,9 A,...,F}
	The :SEARch:SERial:M1553:RTA command specifies the Remote Terminal Address (RTA) value when searching for MIL-STD-1553 events.
<b>Query Syntax</b>	<code>:SEARch:SERial:M1553:RTA?</code>
	The :SEARch:SERial:M1553:RTA? query returns the current Remote Terminal Address value setting.
<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code> <code>&lt;value&gt;</code> ::= 5-bit integer in decimal from 0-31
<b>See Also</b>	<ul style="list-style-type: none"> <li>• <a href="#">Chapter 30</a>, “:SEARch Commands,” starting on page 919</li> </ul>

## :SEARch:SERial:SENT Commands

**Table 134**:SEARch:SERial:SENT Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:SENT:F AST:DATA <string> (see <a href="#">page 999</a> )	:SEARch:SERial:SENT:F AST:DATA? (see <a href="#">page 999</a> )	<string> ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}
:SEARch:SERial:SENT:M ODE <mode> (see <a href="#">page 1000</a> )	:SEARch:SERial:SENT:M ODE? (see <a href="#">page 1000</a> )	<mode> ::= {FCData   SCMid   SCData   CRCerror PPERror}
:SEARch:SERial:SENT:S LOW:DATA <data> (see <a href="#">page 1001</a> )	:SEARch:SERial:SENT:S LOW:DATA? (see <a href="#">page 1001</a> )	<data> ::= from -1 (don't care) to 65535, in NR1 format.
:SEARch:SERial:SENT:S LOW:ID <id> (see <a href="#">page 1002</a> )	:SEARch:SERial:SENT:S LOW:ID? (see <a href="#">page 1002</a> )	<id> ::= from -1 (don't care) to 255, in NR1 format.

**:SEARch:SERial:SENT:FAST:DATA**

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SEARch:SERial:SENT:FAST:DATA &lt;string&gt;</code> <code>&lt;string&gt; ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}</code>
The :SEARch:SERial:SENT:FAST:DATA command specifies the status and data nibbles that will be searched for when the FCData search mode is chosen.	

<b>Query Syntax</b>	<code>:SEARch:SERial:SENT:FAST:DATA?</code>
The :SEARch:SERial:SENT:FAST:DATA? query returns the fast channel data search value setting.	

<b>Return Format</b>	<code>&lt;string&gt;&lt;NL&gt;</code> <code>&lt;string&gt; ::= "0xn..." where n ::= {0,...,9   A,...,F   X   \$}</code>
----------------------	----------------------------------------------------------------------------------------------------------------------------

<b>See Also</b>	<ul style="list-style-type: none"><li><a href="#">":SEARch:SERial:SENT:MODE" on page 1000</a></li><li><a href="#">":SEARch:SERial:SENT:SLOW:DATA" on page 1001</a></li><li><a href="#">":SEARch:SERial:SENT:SLOW:ID" on page 1002</a></li></ul>
-----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :SEARch:SERial:SENT:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:SENT:MODE <mode>`

`<mode> ::= {FCData | SCMid | SCData | CRCerror | PPERror}`

When SENT serial decode is turned on and displayed in the Lister, the :SEARch:SERial:SENT:MODE command specifies what to search for in the decoded data:

- FCData – finds Fast Channel data nibbles that match the values entered using additional softkeys.
- SCMid – finds Slow Channel Message IDs that match the value entered using additional softkeys.
- SCData – finds Slow Channel Message IDs and Data that match the values entered using additional softkeys.
- CRCerror – finds any CRC error, Fast or Slow.
- PPERror – finds where a nibble is either too wide or too narrow (for example, data nibble < 12 (11.5) or > 27 (27.5) ticks wide). Sync, S&C, data, or checksum pulse periods are checked.

**Query Syntax**    `:SEARch:SERial:SENT:MODE?`

The :SEARch:SERial:SENT:MODE? query returns the search mode setting.

**Return Format**    `<mode><NL>`

`<mode> ::= {FCD | SCM | SCD | CRC | PPER}`

**See Also**

- "[":SEARch:SERial:SENT:FAST:DATA](#)" on page 999
- "[":SEARch:SERial:SENT:SLOW:DATA](#)" on page 1001
- "[":SEARch:SERial:SENT:SLOW:ID](#)" on page 1002

**:SEARch:SERial:SENT:SLOW:DATA**

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:SENT:SLOW:DATA <data>

<data> ::= from -1 (don't care) to 65535, in NR1 format.

The :SEARch:SERial:SENT:SLOW:DATA command specifies the data to search for in the Slow Channel Message ID and Data search mode.

**Query Syntax** :SEARch:SERial:SENT:SLOW:DATA?

The :SEARch:SERial:SENT:SLOW:DATA? query returns the slow channel data search value setting.

**Return Format** <data><NL>

<data> ::= from -1 (don't care) to 65535, in NR1 format.

**See Also**

- "[:SEARch:SERial:SENT:FAST:DATA](#)" on page 999
- "[:SEARch:SERial:SENT:MODE](#)" on page 1000
- "[:SEARch:SERial:SENT:SLOW:ID](#)" on page 1002

**:SEARch:SERial:SENT:SLOW:ID****N** (see [page 1334](#))**Command Syntax**    `:SEARch:SERial:SENT:SLOW:ID <id>``<id> ::= from -1 (don't care) to 255, in NR1 format.`

The :SEARch:SERial:SENT:SLOW:ID command specifies the ID to search for in the "Slow Channel Message ID" and "Slow Channel Message ID & Data" trigger modes. The ID can be from -1 (don't care) to 255 (depending on the message length).

**Query Syntax**    `:SEARch:SERial:SENT:SLOW:ID?`

The :SEARch:SERial:SENT:SLOW:ID? query returns the slow channel ID search value setting.

**Return Format**    `<id><NL>``<id> ::= from -1 (don't care) to 255, in NR1 format.`**See Also**

- "[":SEARch:SERial:SENT:FAST:DATA](#)" on page 999
- "[":SEARch:SERial:SENT:MODE](#)" on page 1000
- "[":SEARch:SERial:SENT:SLOW:DATA](#)" on page 1001

## :SEARch:SERial:SPI Commands

**Table 135:** :SEARch:SERial:SPI Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:SPI:MO DE <value> (see <a href="#">page 1004</a> )	:SEARch:SERial:SPI:MO DE? (see <a href="#">page 1004</a> )	<value> ::= {MOSI   MISO}
:SEARch:SERial:SPI:PA TTern:DATA <string> (see <a href="#">page 1005</a> )	:SEARch:SERial:SPI:PA TTern:DATA? (see <a href="#">page 1005</a> )	<string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X}
:SEARch:SERial:SPI:PA TTern:WIDTH <width> (see <a href="#">page 1006</a> )	:SEARch:SERial:SPI:PA TTern:WIDTH? (see <a href="#">page 1006</a> )	<width> ::= integer from 1 to 10

**:SEARch:SERial:SPI:MODE****N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:SPI:MODE <value>`  
                  `<value> ::= {MOSI | MISO}`

The :SEARch:SERial:SPI:MODE command specifies whether the SPI search will be on the MOSI data or the MISO data.

Data values are specified using the :SEARch:SERial:SPI:PATTERn:DATA command.

**Query Syntax**    `:SEARch:SERial:SPI:MODE?`

The :SEARch:SERial:SPI:MODE? query returns the current SPI search mode setting.

**Return Format**    `<value><NL>`  
                  `<value> ::= {MOSI | MISO}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:SPI:PATTERn:DATA"](#) on page 1005

**:SEARch:SERial:SPI:PATTern:DATA**

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:SPI:PATTern:DATA <string>

<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}

The :SEARch:SERial:SPI:PATTern:DATA command specifies the data value when searching for SPI events.

The width of the data value is specified using the :SEARch:SERial:SPI:PATTern:WIDTh command.

**Query Syntax** :SEARch:SERial:SPI:PATTern:DATA?

The :SEARch:SERial:SPI:PATTern:DATA? query returns the current data value setting.

**Return Format** <string><NL>

<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}

**See Also**

- [Chapter 30, “:SEARch Commands,” starting on page 919](#)
- [“:SEARch:SERial:SPI:PATTern:WIDTh” on page 1006](#)

**:SEARch:SERial:SPI:PATTern:WIDTh**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:SPI:PATTern:WIDTh <width>`  
                  `<width> ::= integer from 1 to 10`

The :SEARch:SERial:SPI:PATTern:WIDTh command specifies the width of the data value (in bytes) when searching for SPI events.

The data value is specified using the :SEARch:SERial:SPI:PATTern:DATA command.

**Query Syntax**    `:SEARch:SERial:SPI:PATTern:WIDTh?`

The :SEARch:SERial:SPI:PATTern:WIDTh? query returns the current data width setting.

**Return Format**    `<width><NL>`  
                  `<width> ::= integer from 1 to 10`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919
- [":SEARch:SERial:SPI:PATTern:DATA"](#) on page 1005

## :SEARch:SERial:UART Commands

**Table 136**:SEARch:SERial:UART Commands Summary

Command	Query	Options and Query Returns
:SEARch:SERial:UART:D ATA <value> (see <a href="#">page 1008</a> )	:SEARch:SERial:UART:D ATA? (see <a href="#">page 1008</a> )	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, <binary>, or <quoted_string> format <hexadecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0   1} for binary <quoted_string> ::= any of the 128 valid 7-bit ASCII characters (or standard abbreviations)
:SEARch:SERial:UART:M ODE <value> (see <a href="#">page 1009</a> )	:SEARch:SERial:UART:M ODE? (see <a href="#">page 1009</a> )	<value> ::= {RDATa   RD1   RD0   RDX   TDATA   TD1   TD0   TDX   PARityerror   AERRor}
:SEARch:SERial:UART:Q UALifier <value> (see <a href="#">page 1010</a> )	:SEARch:SERial:UART:Q UALifier? (see <a href="#">page 1010</a> )	<value> ::= {EQUAL   NOTEqual   GREaterthan   LESSthan}

## :SEARch:SERial:UART:DATA

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:SEARch:SERial:UART:DATA &lt;value&gt;</code>
	<code>&lt;value&gt;</code> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <code>&lt;hexadecimal&gt;, &lt;binary&gt;, or &lt;quoted_string&gt; format</code>
	<code>&lt;hexadecimal&gt;</code> ::= #Hnn where n ::= {0,...,9  A,...,F} for hexadecimal
	<code>&lt;binary&gt;</code> ::= #Bnn...n where n ::= {0   1} for binary
	<code>&lt;quoted_string&gt;</code> ::= any of the 128 valid 7-bit ASCII characters (or standard abbreviations)

The :SEARch:SERial:UART:DATA command specifies a data value when searching for UART/RS232 events.

The data value qualifier is specified using the :SEARch:SERial:UART:QUALifier command.

<b>Query Syntax</b>	<code>:SEARch:SERial:UART:DATA?</code>
	The :SEARch:SERial:UART:DATA? query returns the current data value setting.

<b>Return Format</b>	<code>&lt;value&gt;&lt;NL&gt;</code>
	<code>&lt;value&gt;</code> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal format

<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">Chapter 30</a>, “:SEARch Commands,” starting on page 919</li> <li>· <a href="#">":SEARch:SERial:UART:MODE"</a> on page 1009</li> <li>· <a href="#">":SEARch:SERial:UART:QUALifier"</a> on page 1010</li> </ul>
-----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :SEARch:SERial:UART:MODE

**N** (see [page 1334](#))

**Command Syntax** :SEARch:SERial:UART:MODE <value>

```
<value> ::= {RDATA | RD1 | RD0 | RDX | TDATA | TD1 | TD0 | TDX
             | PARityerror | AERRor}
```

The :SEARch:SERial:UART:MODE command selects the type of UART/RS232 information to find in the Lister display:

- RDATA – searches for a receive data value when data words are from 5 to 8 bits long.
- RD1 – searches for a receive data value when data words are 9 bits long and the 9th (alert) bit is 1.
- RD0 – searches for a receive data value when data words are 9 bits long and the 9th (alert) bit is 0.
- RDX – searches for a receive data value when data words are 9 bits long and the 9th (alert) bit is a don't care (X).
- TDATA – searches for a transmit data value when data words are from 5 to 8 bits long.
- TD1 – searches for a transmit data value when data words are 9 bits long and the 9th (alert) bit is 1.
- TD0 – searches for a transmit data value when data words are 9 bits long and the 9th (alert) bit is 0.
- TDX – searches for a transmit data value when data words are 9 bits long and the 9th (alert) bit is a don't care (X).
- PARityerror – searches for parity errors.
- AERRor – searches for any error.

Data values are specified using the :SEARch:SERial:UART:DATA command.

Data value qualifiers are specified using the :SEARch:SERial:UART:QUALifier command.

**Query Syntax** :SEARch:SERial:UART:MODE?

The :SEARch:SERial:UART:MODE? query returns ...

**Return Format** <value><NL>

```
<value> ::= {RDAT | RD1 | RD0 | RDX | TDAT | TD1 | TD0 | TDX | PAR
             | AERR}
```

- See Also**
- [Chapter 30](#), “:SEARch Commands,” starting on page 919
  - [":SEARch:SERial:UART:DATA"](#) on page 1008
  - [":SEARch:SERial:UART:QUALifier"](#) on page 1010

**:SEARch:SERial:UART:QUALifier**

**N** (see [page 1334](#))

**Command Syntax**    `:SEARch:SERial:UART:QUALifier <value>`

`<value> ::= {EQUAL | NOTEQUAL | GREATERthan | LESSthan}`

The :SEARch:SERial:UART:QUALifier command specifies the data value qualifier when searching for UART/RS232 events.

**Query Syntax**    `:SEARch:SERial:UART:QUALifier?`

The :SEARch:SERial:UART:QUALifier? query returns the current data value qualifier setting.

**Return Format**    `<value><NL>`

`<value> ::= {EQU | NOT | GRE | LESS}`

**See Also**

- [Chapter 30](#), “:SEARch Commands,” starting on page 919

- [":SEARch:SERial:UART:DATA"](#) on page 1008

# 31 :SYSTem Commands

Control basic system functions of the oscilloscope. See "[Introduction to :SYSTem Commands](#)" on page 1012.

**Table 137:**:SYSTem Commands Summary

Command	Query	Options and Query Returns
:SYSTem:DATE <date> (see <a href="#">page 1013</a> )	:SYSTem:DATE? (see <a href="#">page 1013</a> )	<date> ::= <year>,<month>,<day> <year> ::= 4-digit year in NR1 format <month> ::= {1,...,12   JANuary   FEBruary   MARch   APRil   MAY   JUNe   JULy   AUGust   SEPtember   OCTober   NOVember   DECember} <day> ::= {1,...31}
:SYSTem:DSP <string> (see <a href="#">page 1014</a> )	n/a	<string> ::= up to 75 characters as a quoted ASCII string
n/a	:SYSTem:ERRor? (see <a href="#">page 1015</a> )	<error> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see <a href="#">page 1291</a> ).
:SYSTem:LOCK <value> (see <a href="#">page 1016</a> )	:SYSTem:LOCK? (see <a href="#">page 1016</a> )	<value> ::= {{1   ON}   {0   OFF}}
:SYSTem:PERSONa [:MANufacturer] <manufacturer_string> (see <a href="#">page 1017</a> )	:SYSTem:PERSONa [:MANufacturer]? (see <a href="#">page 1017</a> )	<manufacturer_string> ::= quoted ASCII string, up to 63 characters
:SYSTem:PERSONa [:MANufacturer] :DEFault (see <a href="#">page 1018</a> )	n/a	Sets manufacturer string to "KEYSIGHT TECHNOLOGIES"
:SYSTem:PRESet (see <a href="#">page 1019</a> )	n/a	See :SYSTem:PRESet (see <a href="#">page 1019</a> )

**Table 137:**SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:PROTection:LOCK <value> (see page 1022)	:SYSTem:PROTection:LOCK? (see page 1022)	<value> ::= {{1   ON}   {0   OFF}}
:SYSTem:RLOGger <setting>[,<file_name>[,<write_mode>]] (see page 1023)	n/a	<setting> ::= {0   OFF}   {1   ON} <file_name> ::= quoted ASCII string <write_mode> ::= {CREATE   APPEND}
:SYSTem:RLOGger:DESTination <dest> (see page 1024)	:SYSTem:RLOGger:DESTination? (see page 1024)	<dest> ::= {FILE   SCReen   BOTH}
:SYSTem:RLOGger:DISPLAY {{0   OFF}   {1   ON}} (see page 1025)	:SYSTem:RLOGger:DISPLAY? (see page 1025)	<setting> ::= {0   1}
:SYSTem:RLOGger:FNAMe <file_name> (see page 1026)	:SYSTem:RLOGger:FNAMe? (see page 1026)	<file_name> ::= quoted ASCII string
:SYSTem:RLOGger:STATE {{0   OFF}   {1   ON}} (see page 1027)	:SYSTem:RLOGger:STATE? (see page 1027)	<setting> ::= {0   1}
:SYSTem:RLOGger:TRANSparent {{0   OFF}   {1   ON}} (see page 1028)	:SYSTem:RLOGger:TRANSparent? (see page 1028)	<setting> ::= {0   1}
:SYSTem:RLOGger:WMODe <write_mode> (see page 1029)	:SYSTem:RLOGger:WMODe? (see page 1029)	<write_mode> ::= {CREATE   APPEND}
:SYSTem:SETup <setup_data> (see page 1030)	:SYSTem:SETup? (see page 1030)	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:TIME <time> (see page 1032)	:SYSTem:TIME? (see page 1032)	<time> ::= hours,minutes,seconds in NR1 format
:SYSTem:TOUCH {{1   ON}   {0   OFF}} (see page 1033)	:SYSTem:TOUCH? (see page 1033)	{1   0}

**Introduction to :SYSTem Commands** SYSTem subsystem commands enable writing messages to the display, setting and reading both the time and the date, querying for errors, and saving and recalling setups.

## :SYSTem:DATE

**N** (see [page 1334](#))

**Command Syntax** :SYSTem:DATE <date>

```
<date> ::= <year>,<month>,<day>
<year> ::= 4-digit year in NR1 format
<month> ::= {1,...,12 | JANuary | FEBruary | MARch | APRil | MAY | JUNE
              | JULY | AUGust | SEPtember | OCTober | NOVember | DECember}
<day> ::= {1,...,31}
```

The :SYSTem:DATE command sets the date. Validity checking is performed to ensure that the date is valid.

**Query Syntax** :SYSTem:DATE?

The SYSTem:DATE? query returns the date.

**Return Format** <year>,<month>,<day><NL>

**See Also** • "[Introduction to :SYSTem Commands](#)" on page 1012  
• "[":SYSTem:TIME"](#) on page 1032

## :SYSTem:DSP

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:DSP <string>`

`<string> ::= quoted ASCII string (up to 75 characters)`

The :SYSTem:DSP command writes the quoted string (excluding quotation marks) to a text box on-screen.

Use :SYSTem:DSP "" to remotely remove the message from the display. (Two sets of quote marks without a space between them creates a NULL string.)

Press any menu key to manually remove the message from the display.

**See Also**    • "Introduction to :SYSTem Commands" on page 1012

## :SYSTem:ERRor



(see [page 1334](#))

### Query Syntax :SYSTem:ERRor?

The :SYSTem:ERRor? query outputs the next error number and text from the error queue. The instrument has an error queue that is 30 errors deep and operates on a first-in, first-out basis. Repeatedly sending the :SYSTem:ERRor? query returns the errors in the order that they occurred until the queue is empty. Any further queries then return zero until another error occurs.

When remote logging is enabled (using the oscilloscope's front panel), additional debug information can be included in the returned error string. If the error is detected by the SCPI command parser, such as a header error or other syntax error, the extra debug information is generated and included. But if the error is detected by the oscilloscope system, such as when an out-of-range value is sent, then no extra debug information is included.

<b>Return Format</b>	<pre>&lt;error number&gt;,&lt;error string&gt;&lt;NL&gt;</pre> <p>&lt;error number&gt; ::= an integer error code in NR1 format</p> <p>&lt;error string&gt; ::= quoted ASCII string containing the error message</p> <p>Error messages are listed in <a href="#">Chapter 38</a>, “Error Messages,” starting on page 1291.</p>
<b>See Also</b>	<ul style="list-style-type: none"> <li>• <a href="#">"Introduction to :SYSTem Commands"</a> on page 1012</li> <li>• <a href="#">"*ESR (Standard Event Status Register)"</a> on page 184</li> <li>• <a href="#">"*CLS (Clear Status)"</a> on page 181</li> </ul>

## :SYSTem:LOCK

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:LOCK <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :SYSTem:LOCK command disables the front panel. LOCK ON is the equivalent of sending a local lockout message over the programming interface.

**Query Syntax**    `:SYSTem:LOCK?`

The :SYSTem:LOCK? query returns the lock status of the front panel.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**    • "Introduction to :SYSTem Commands" on page 1012

**:SYSTem:PERSONa[:MANufacturer]****N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:PERSONa [:MANufacturer] <manufacturer_string>`  
`<manufacturer_string> ::= ::= quoted ASCII string, up to 63 characters`

The :SYSTem:PERSONa[:MANufacturer] command lets you change the manufacturer string portion of the identification string returned by the \*IDN? query.

The default manufacturer string is "KEYSIGHT TECHNOLOGIES".

If your remote programs depend on a legacy manufacturer string, for example, you could use this command to set the manufacturer string to "AGILENT TECHNOLOGIES".

**Query Syntax**    `:SYSTem:PERSONa [:MANufacturer]?`

The :SYSTem:PERSONa[:MANufacturer]? query returns the currently set manufacturer string.

**Return Format**    `<manufacturer_string><NL>`

**See Also**

- ["\\*IDN \(Identification Number\)" on page 186](#)
- [":SYSTem:PERSONa\[:MANufacturer\]:DEFault" on page 1018](#)
- ["Introduction to :SYSTem Commands" on page 1012](#)

## :SYSTem:PERSONa[:MANufacturer]:DEFault

**N** (see [page 1334](#))

**Command Syntax** :SYSTem:PERSONa [:MANufacturer] :DEFault

The :SYSTem:PERSONa[:MANufacturer]:DEFault command sets the manufacturer string to "KEYSIGHT TECHNOLOGIES".

**See Also**

- ["\\*IDN \(Identification Number\)" on page 186](#)
- [":SYSTem:PERSONa\[:MANufacturer\]" on page 1017](#)
- ["Introduction to :SYSTem Commands" on page 1012](#)

## :SYSTem:PRESet

**C** (see [page 1334](#))

**Command Syntax** :SYSTem:PRESet

The :SYSTem:PRESet command places the instrument in a known state. This is the same as pressing the **[Default Setup]** key or **[Save/Recall] > Default/Erase > Default Setup** on the front panel.

When you perform a default setup, some user settings (like preferences) remain unchanged. To reset all user settings to their factory defaults, use the \*RST command.

Reset conditions are:

Acquire Menu	
Mode	Normal
Averaging	Off
# Averages	8

Analog Channel Menu	
Channel 1	On
Channel 2	Off
Volts/division	5.00 V
Offset	0.00
Coupling	DC
Probe attenuation	10:1
Vernier	Off
Invert	Off
BW limit	Off
Impedance	1 M Ohm (cannot be changed)
Units	Volts
Skew	0

Cursor Menu	
Source	Channel 1

<b>Digital Channel Menu (MSO models only)</b>	
Channel 0 - 7	Off
Labels	Off
Threshold	TTL (1.4 V)

<b>Display Menu</b>	
Persistence	Off
Grid	20%

<b>Quick Meas Menu</b>	
Source	Channel 1

<b>Run Control</b>	
	Scope is running

<b>Time Base Menu</b>	
Main time/division	100 us
Main time base delay	0.00 s
Delay time/division	500 ns
Delay time base delay	0.00 s
Reference	center
Mode	main
Vernier	Off

<b>Trigger Menu</b>	
Type	Edge
Mode	Auto
Coupling	dc
Source	Channel 1
Level	0.0 V
Slope	Positive

Trigger Menu	
HF Reject and noise reject	Off
Holdoff	40 ns
External probe attenuation	10:1
External Units	Volts
External Impedance	1 M Ohm (cannot be changed)

- See Also
- ["Introduction to Common \(\\*\) Commands"](#) on page 180
  - ["\\*RST \(Reset\)"](#) on page 192

**:SYSTem:PROTection:LOCK**

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:PROTection:LOCK <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :SYSTem:PROTection:LOCK command disables the fifty ohm impedance setting for all analog channels.

**Query Syntax**    `:SYSTem:PROTection:LOCK?`

The :SYSTem:PROTection:LOCK? query returns the analog channel protection lock status.

**Return Format**    `<on_off><NL>`  
`<on_off> ::= {1 | 0}`

**See Also**    · ["Introduction to :SYSTem Commands"](#) on page 1012

## :SYSTem:RLOGger

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:RLOGger <setting>[,<file_name>[,<write_mode>]]`  
`<setting> ::= {{0 | OFF} | {1 | ON}}`  
`<file_name> ::= quoted ASCII string`  
`<write_mode> ::= {CREATE | APPEND}`

The :SYSTem:RLOGger command enables or disables remote command logging, optionally specifying the log file name and write mode.

- See Also**
- "[:SYSTem:RLOGger:DESTination](#)" on page 1024
  - "[:SYSTem:RLOGger:DISPLAY](#)" on page 1025
  - "[:SYSTem:RLOGger:FNAMe](#)" on page 1026
  - "[:SYSTem:RLOGger:STATE](#)" on page 1027
  - "[:SYSTem:RLOGger:TRANsparent](#)" on page 1028
  - "[:SYSTem:RLOGger:WMODE](#)" on page 1029

## :SYSTem:RLOGger:DESTination

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:RLOGger:DESTination <dest>`  
`<dest> ::= {FILE | SCReen | BOTH}`

The :SYSTem:RLOGger:DESTination command specifies whether remote commands are logged to a text file (on a connected USB storage device), logged to the screen, or both.

**NOTE** If the destination is changed while remote command logging is running, remote command logging is turned off.

---

**Query Syntax**    `:SYSTem:RLOGger:DESTination?`

The :SYSTem:RLOGger:DESTination? query returns the remote command logging destination.

**Return Format**    `<dest><NL>`

`<dest> ::= {FILE | SCR | BOTH}`

**See Also**

- "[:SYSTem:RLOGger](#)" on page 1023
- "[:SYSTem:RLOGger:DISPlay](#)" on page 1025
- "[:SYSTem:RLOGger:FNAME](#)" on page 1026
- "[:SYSTem:RLOGger:STATE](#)" on page 1027
- "[:SYSTem:RLOGger:TRANsparent](#)" on page 1028
- "[:SYSTem:RLOGger:WMODE](#)" on page 1029

## :SYSTem:RLOGger:DISPlay

**N** (see [page 1334](#))

**Command Syntax** :SYSTem:RLOGger:DISPlay {{0 | OFF} | {1 | ON}}

The :SYSTem:RLOGger:DISPlay command enables or disables the screen display of logged remote commands and their return values (if applicable).

**Query Syntax** :SYSTem:RLOGger:DISPlay?

The :SYSTem:RLOGger:DISPlay? query returns whether the screen display for remote command logging is enabled or disabled.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

**See Also**

- "[:SYSTem:RLOGger](#)" on page 1023
- "[:SYSTem:RLOGger:DESTination](#)" on page 1024
- "[:SYSTem:RLOGger:FNAME](#)" on page 1026
- "[:SYSTem:RLOGger:STATE](#)" on page 1027
- "[:SYSTem:RLOGger:TRANsparent](#)" on page 1028
- "[:SYSTem:RLOGger:WMODE](#)" on page 1029

**:SYSTem:RLOGger:FNAME**

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:RLOGger:FNAME <file_name>`  
`<file_name> ::= quoted ASCII string`

The :SYSTem:RLOGger:FNAME command specifies the remote command log file name.

Because log files are ASCII text files, the ".txt" extension is automatically added to the name specified.

**Query Syntax**    `:SYSTem:RLOGger:FNAME?`

The :SYSTem:RLOGger:FNAME? query returns the remote command log file name.

**Return Format**    `<file_name><NL>`

**See Also**

- "[":SYSTem:RLOGger"](#) on page 1023
- "[":SYSTem:RLOGger:DESTination"](#) on page 1024
- "[":SYSTem:RLOGger:DISPLAY"](#) on page 1025
- "[":SYSTem:RLOGger:STATE"](#) on page 1027
- "[":SYSTem:RLOGger:TRANsparent"](#) on page 1028
- "[":SYSTem:RLOGger:WMODE"](#) on page 1029

## :SYSTem:RLOGger:STATE

**N** (see [page 1334](#))

**Command Syntax** :SYSTem:RLOGger:STATE {{0 | OFF} | {1 | ON}}

The :SYSTem:RLOGger:STATE command enables or disables remote command logging.

**Query Syntax** :SYSTem:RLOGger:STATE?

The :SYSTem:RLOGger:STATE? query returns the remote command logging state.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- "[:SYSTem:RLOGger](#)" on page 1023
  - "[:SYSTem:RLOGger:DESTination](#)" on page 1024
  - "[:SYSTem:RLOGger:DISPLAY](#)" on page 1025
  - "[:SYSTem:RLOGger:FNAMe](#)" on page 1026
  - "[:SYSTem:RLOGger:TRANsparent](#)" on page 1028
  - "[:SYSTem:RLOGger:WMODE](#)" on page 1029

## :SYSTem:RLOGger:TRANsparent

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:RLOGger:TRANsparent {{0 | OFF} | {1 | ON}}`

The :SYSTem:RLOGger:TRANsparent command specifies whether the screen display background for remote command logging is transparent or solid.

**Query Syntax**    `:SYSTem:RLOGger:TRANsparent?`

The :SYSTem:RLOGger:TRANsparent? query returns the setting for transparent screen display background.

**Return Format**    `<setting><NL>`

`<setting> ::= {0 | 1}`

**See Also**

- "[:SYSTem:RLOGger](#)" on page 1023
- "[:SYSTem:RLOGger:DESTination](#)" on page 1024
- "[:SYSTem:RLOGger:DISPLAY](#)" on page 1025
- "[:SYSTem:RLOGger:FNAME](#)" on page 1026
- "[:SYSTem:RLOGger:STATE](#)" on page 1027
- "[:SYSTem:RLOGger:WMODe](#)" on page 1029

**:SYSTem:RLOGger:WMODE**

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:RLOGger:WMODE <write_mode>`  
`<write_mode> ::= {CREate | APPend}`

The :SYSTem:RLOGger:WMODE command specifies the remote command logging write mode.

**Query Syntax**    `:SYSTem:RLOGger:WMODE?`

The :SYSTem:RLOGger:WMODE? query returns the remote command logging write mode.

**Return Format**    `<write_mode><NL>`  
`<write_mode> ::= {CRE | APP}`

**See Also**

- [":SYSTem:RLOGger"](#) on page 1023
- [":SYSTem:RLOGger:DESTination"](#) on page 1024
- [":SYSTem:RLOGger:DISPLAY"](#) on page 1025
- [":SYSTem:RLOGger:FNAMe"](#) on page 1026
- [":SYSTem:RLOGger:STATE"](#) on page 1027
- [":SYSTem:RLOGger:TRANsparent"](#) on page 1028

## :SYSTem:SEtup

 (see [page 1334](#))

<b>Command Syntax</b>	<code>:SYSTem:SEtup &lt;setup_data&gt;</code> <code>&lt;setup_data&gt; ::= binary block data in IEEE 488.2 # format.</code>
	The :SYSTem:SEtup command sets the oscilloscope as defined by the data in the setup (learn) string sent from the controller. The setup string does not change the interface mode or interface address.
<b>Query Syntax</b>	<code>:SYSTem:SEtup?</code>
	The :SYSTem:SEtup? query operates the same as the *LRN? query. It outputs the current oscilloscope setup in the form of a learn string to the controller. The setup (learn) string is sent and received as a binary block of data. The format for the data transmission is the # format defined in the IEEE 488.2 specification.
<b>Return Format</b>	<code>&lt;setup_data&gt;&lt;NL&gt;</code> <code>&lt;setup_data&gt; ::= binary block data in IEEE 488.2 # format</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">"Introduction to :SYSTem Commands"</a> on page 1012</li> <li>· <a href="#">"*LRN (Learn Device Setup)"</a> on page 187</li> </ul>
<b>Example Code</b>	<pre> ' SAVE_SYSTEM_SETUP - The :SYSTEM:SETUP? query returns a program ' message that contains the current state of the instrument. Its ' format is a definite-length binary block, for example, ' #800075595&lt;setup string&gt;&lt;NL&gt; ' where the setup string is 75595 bytes in length. myScope.WriteString ":SYSTEM:SETUP?" varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1) CheckForInstrumentErrors      ' After reading query results.  ' Output setup string to a file: Dim strPath As String strPath = "c:\scope\config\setup.dat"  ' Open file for output. Close #1      ' If #1 is open, close it. Open strPath For Binary Access Write Lock Write As #1 Put #1, , varQueryResult      ' Write data. Close #1      ' Close file.  ' RESTORE_SYSTEM_SETUP - Read the setup string from a file and ' write it back to the oscilloscope. Dim varSetupString As Variant strPath = "c:\scope\config\setup.dat"  ' Open file for input. Open strPath For Binary Access Read As #1 Get #1, , varSetupString      ' Read data. Close #1      ' Close file.  ' Write setup string back to oscilloscope using ":SYSTEM:SETUP" </pre>

```
' command:  
myScope.WriteIEEEBlock ":SYSTEM:SETUP ", varSetupString  
CheckForInstrumentErrors
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

**:SYSTem:TIME**

**N** (see [page 1334](#))

**Command Syntax**    `:SYSTem:TIME <time>`

`<time> ::= hours,minutes,seconds in NR1 format`

The :SYSTem:TIME command sets the system time, using a 24-hour format. Commas are used as separators. Validity checking is performed to ensure that the time is valid.

**Query Syntax**    `:SYSTem:TIME? <time>`

The :SYSTem:TIME? query returns the current system time.

**Return Format**    `<time><NL>`

`<time> ::= hours,minutes,seconds in NR1 format`

**See Also**

- "[Introduction to :SYSTem Commands](#)" on page 1012

- "[":SYSTem:DATE](#)" on page 1013

## :SYSTem:TOUCH

**N** (see [page 1334](#))

**Command Syntax** :SYSTem:TOUCH <on\_off>

<on\_off> ::= {{1 | ON} | {0 | OFF}}

The :SYSTem:TOUCH command disables or enables the touchscreen.

**Query Syntax** :SYSTem:TOUCH?

The :SYSTem:TOUCH? query returns the touchscreen's on/off status.

**Return Format** <on\_off><NL>

<on\_off> ::= {1 | 0}

**See Also** · "Introduction to :SYSTem Commands" on page 1012



## 32 :TIMEbase Commands

Control all horizontal sweep functions. See "[Introduction to :TIMEbase Commands](#)" on page 1036.

**Table 138:**TIMEbase Commands Summary

Command	Query	Options and Query Returns
:TIMEbase:MODE <value> (see <a href="#">page 1037</a> )	:TIMEbase:MODE? (see <a href="#">page 1037</a> )	<value> ::= {MAIN   WINDOW   XY   ROLL}
:TIMEbase:POSITION <pos> (see <a href="#">page 1038</a> )	:TIMEbase:POSITION? (see <a href="#">page 1038</a> )	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMEbase:RANGE <range_value> (see <a href="#">page 1039</a> )	:TIMEbase:RANGE? (see <a href="#">page 1039</a> )	<range_value> ::= time for 10 div in seconds in NR3 format
:TIMEbase:REFERENCE {LEFT   CENTER   RIGHT} (see <a href="#">page 1040</a> )	:TIMEbase:REFERENCE? (see <a href="#">page 1040</a> )	<return_value> ::= {LEFT   CENTER   RIGHT}
:TIMEbase:SCALE <scale_value> (see <a href="#">page 1041</a> )	:TIMEbase:SCALE? (see <a href="#">page 1041</a> )	<scale_value> ::= time/div in seconds in NR3 format
:TIMEbase:VERNier {{0   OFF}   {1   ON}} (see <a href="#">page 1042</a> )	:TIMEbase:VERNier? (see <a href="#">page 1042</a> )	{0   1}
:TIMEbase:WINDOW:POSITION <pos> (see <a href="#">page 1043</a> )	:TIMEbase:WINDOW:POSITION? (see <a href="#">page 1043</a> )	<pos> ::= time from the trigger event to the zoomed view reference point in NR3 format
:TIMEbase:WINDOW:RANGE <range_value> (see <a href="#">page 1044</a> )	:TIMEbase:WINDOW:RANGE? (see <a href="#">page 1044</a> )	<range value> ::= range value in seconds in NR3 format for the zoomed window
:TIMEbase:WINDOW:SCALE <scale_value> (see <a href="#">page 1045</a> )	:TIMEbase:WINDOW:SCALE? (see <a href="#">page 1045</a> )	<scale_value> ::= scale value in seconds in NR3 format for the zoomed window

**Introduction to :T1Mebase Commands** The T1Mebase subsystem commands control the horizontal (X-axis) functions and set the oscilloscope to X-Y mode (where channel 1 becomes the X input and channel 2 becomes the Y input). The time per division, delay, vernier control, and reference can be controlled for the main and window (zoomed) time bases.

### Reporting the Setup

Use :T1Mebase? to query setup information for the T1Mebase subsystem.

### Return Format

The following is a sample response from the :T1Mebase? query. In this case, the query was issued following a \*RST command.

```
:T1M:MODE MAIN;REF CENT;MAIN:RANG +1.00E-03;POS +0.0E+00
```

## :TIMEbase:MODE

 (see [page 1334](#))

**Command Syntax**    `:TIMEbase:MODE <value>`

`<value> ::= {MAIN | WINDow | XY | ROLL}`

The :TIMEbase:MODE command sets the current time base. There are four time base modes:

- MAIN – The normal time base mode is the main time base. It is the default time base mode after the \*RST (Reset) command.
- WINDow – In the WINDow (zoomed or delayed) time base mode, measurements are made in the zoomed time base if possible; otherwise, the measurements are made in the main time base.
- XY – In the XY mode, the :TIMEbase:RANGE, :TIMEbase:POSITION, and :TIMEbase:REFERENCE commands are not available. No measurements are available in this mode.
- ROLL – In the ROLL mode, data moves continuously across the display from left to right. The oscilloscope runs continuously and is untriggered. The :TIMEbase:REFERENCE selection changes to RIGHT.

**Query Syntax**    `:TIMEbase:MODE?`

The :TIMEbase:MODE query returns the current time base mode.

**Return Format**    `<value><NL>`

`<value> ::= {MAIN | WIND | XY | ROLL}`

**See Also**

- "[Introduction to :TIMEbase Commands](#)" on page 1036
- "[\\*RST \(Reset\)](#)" on page 192
- "[:TIMEbase:RANGE](#)" on page 1039
- "[:TIMEbase:POSITION](#)" on page 1038
- "[:TIMEbase:REFERENCE](#)" on page 1040

**Example Code**

```
' TIMEBASE_MODE - (not executed in this example)
' Set the time base mode to MAIN, DELAYED, XY, or ROLL.

' Set time base mode to main.
myScope.WriteString ":TIMEBASE:MODE MAIN"
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

## :TIMEbase:POSITION

 (see [page 1334](#))

**Command Syntax**    `:TIMEbase:POSITION <pos>`

`<pos>` ::= time in seconds from the trigger to the display reference  
in NR3 format

The :TIMEbase:POSITION command sets the time interval between the trigger event and the display reference point on the screen. The display reference point is either left, right, or center and is set with the :TIMEbase:REFERENCE command. The maximum position value depends on the time/division settings.

### NOTE

This command is an alias for the :TIMEbase:DELay command.

**Query Syntax**    `:TIMEbase:POSITION?`

The :TIMEbase:POSITION? query returns the current time from the trigger to the display reference in seconds.

**Return Format**    `<pos><NL>`

`<pos>` ::= time in seconds from the trigger to the display reference  
in NR3 format

**See Also**

- "[Introduction to :TIMEbase Commands](#)" on page 1036
- "[:TIMEbase:REFERENCE](#)" on page 1040
- "[:TIMEbase:RANGE](#)" on page 1039
- "[:TIMEbase:SCALE](#)" on page 1041
- "[:TIMEbase:WINDOW:POSITION](#)" on page 1043
- "[:TIMEbase:DELay](#)" on page 1287

## :TIMEbase:RANGE

**C** (see [page 1334](#))

**Command Syntax** :TIMEbase:RANGE <range\_value>

<range\_value> ::= time for 10 div in seconds in NR3 format

The :TIMEbase:RANGE command sets the full-scale horizontal time in seconds for the main window. The range is 10 times the current time-per-division setting.

**Query Syntax** :TIMEbase:RANGE?

The :TIMEbase:RANGE query returns the current full-scale range value for the main window.

**Return Format** <range\_value><NL>

<range\_value> ::= time for 10 div in seconds in NR3 format

- See Also**
- "[Introduction to :TIMEbase Commands](#)" on page 1036
  - "[:TIMEbase:MODE](#)" on page 1037
  - "[:TIMEbase:SCALe](#)" on page 1041
  - "[:TIMEbase:WINDOW:RANGE](#)" on page 1044

**Example Code**

```
' TIME_RANGE - Sets the full scale horizontal time in seconds.  The
' range value is 10 times the time per division.
myScope.WriteString ":TIM:RANG 2e-3"    ' Set the time range to 0.002
seconds.
```

See complete example programs at: [Chapter 42](#), "Programming Examples," starting on page 1343

## :TImebase:REFerence



(see [page 1334](#))

**Command Syntax**

```
:TImebase:REFerence <reference>
<reference> ::= {LEFT | CENTer | RIGHT}
```

The :TImebase:REFerence command sets the time reference to one division from the left side of the screen, to the center of the screen, or to one division from the right side of the screen. Time reference is the point on the display where the trigger point is referenced.

**Query Syntax**

```
:TImebase:REFerence?
```

The :TImebase:REFerence? query returns the current display reference for the main window.

**Return Format**

```
<reference><NL>
<reference> ::= {LEFT | CENT | RIGH}
```

**See Also**

- ["Introduction to :TImebase Commands"](#) on page 1036
- [":TImebase:MODE"](#) on page 1037

**Example Code**

```
' TIME_REFERENCE - Possible values are LEFT, CENTer, or RIGHT.
'   - LEFT sets the display reference one time division from the left.
'   - CENTer sets the display reference to the center of the screen.
'   - RIGHT sets the display reference one time division from the righ
t.
myScope.WriteString ":TImebase:REFerence CENTER"      ' Set reference to
center.
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

**:TIMEbase:SCALe**

**N** (see [page 1334](#))

**Command Syntax** :TIMEbase:SCALe <scale\_value>

<scale\_value> ::= time/div in seconds in NR3 format

The :TIMEbase:SCALe command sets the horizontal scale or units per division for the main window.

**Query Syntax** :TIMEbase:SCALe?

The :TIMEbase:SCALe? query returns the current horizontal scale setting in seconds per division for the main window.

**Return Format** <scale\_value><NL>

<scale\_value> ::= time/div in seconds in NR3 format

**See Also**

- "[Introduction to :TIMEbase Commands](#)" on page 1036
- "[:TIMEbase:RANGE](#)" on page 1039
- "[:TIMEbase:WINDOW:SCALe](#)" on page 1045
- "[:TIMEbase:WINDOW:RANGE](#)" on page 1044

**:TIMEbase:VERNier**

**N** (see [page 1334](#))

**Command Syntax**    `:TIMEbase:VERNier <vernier value>`

`<vernier value> ::= {{1 | ON} | {0 | OFF}}`

The :TIMEbase:VERNier command specifies whether the time base control's vernier (fine horizontal adjustment) setting is ON (1) or OFF (0).

**Query Syntax**    `:TIMEbase:VERNier?`

The :TIMEbase:VERNier? query returns the current state of the time base control's vernier setting.

**Return Format**    `<vernier value><NL>`

`<vernier value> ::= {0 | 1}`

**See Also**    · "Introduction to :TIMEbase Commands" on page 1036

## :TIMEbase:WINDOW:POSITION



(see [page 1334](#))

**Command Syntax**    `:TIMEbase:WINDOW:POSITION <pos value>`

`<pos value>` ::= time from the trigger event to the zoomed (delayed) view reference point in NR3 format

The :TIMEbase:WINDOW:POSITION command sets the horizontal position in the zoomed (delayed) view of the main sweep. The main sweep range and the main sweep horizontal position determine the range for this command. The value for this command must keep the zoomed view window within the main sweep range.

**Query Syntax**    `:TIMEbase:WINDOW:POSITION?`

The :TIMEbase:WINDOW:POSITION? query returns the current horizontal window position setting in the zoomed view.

**Return Format**    `<value><NL>`

`<value>` ::= position value in seconds

- See Also**
- "Introduction to :TIMEbase Commands" on page 1036
  - "[:TIMEbase:MODE](#)" on page 1037
  - "[:TIMEbase:POStion](#)" on page 1038
  - "[:TIMEbase:RANGE](#)" on page 1039
  - "[:TIMEbase:SCALe](#)" on page 1041
  - "[:TIMEbase:WINDOW:RANGE](#)" on page 1044
  - "[:TIMEbase:WINDOW:SCALe](#)" on page 1045

**:TIMEbase:WINDOW:RANGE****C** (see [page 1334](#))**Command Syntax**    `:TIMEbase:WINDOW:RANGE <range value>``<range value> ::= range value in seconds in NR3 format`

The :TIMEbase:WINDOW:RANGE command sets the full-scale horizontal time in seconds for the zoomed (delayed) window. The range is 10 times the current zoomed view window seconds per division setting. The main sweep range determines the range for this command. The maximum value is one half of the :TIMEbase:RANGE value.

**Query Syntax**    `:TIMEbase:WINDOW:RANGE?`

The :TIMEbase:WINDOW:RANGE? query returns the current window timebase range setting.

**Return Format**    `<value><NL>``<value> ::= range value in seconds`**See Also**

- "[Introduction to :TIMEbase Commands](#)" on page 1036
- "[:TIMEbase:RANGE](#)" on page 1039
- "[:TIMEbase:POSIon](#)" on page 1038
- "[:TIMEbase:SCALe](#)" on page 1041

## :TIMEbase:WINDOW:SCALe

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:TIMEbase:WINDOW:SCALe &lt;scale_value&gt;</code> <code>&lt;scale_value&gt; ::= scale value in seconds in NR3 format</code>
	The :TIMEbase:WINDOW:SCALe command sets the zoomed (delayed) window horizontal scale (seconds/division). The main sweep scale determines the range for this command. The maximum value is one half of the :TIMEbase:SCALe value.
<b>Query Syntax</b>	<code>:TIMEbase:WINDOW:SCALe?</code>
	The :TIMEbase:WINDOW:SCALe? query returns the current zoomed window scale setting.
<b>Return Format</b>	<code>&lt;scale_value&gt;&lt;NL&gt;</code> <code>&lt;scale_value&gt; ::= current seconds per division for the zoomed window</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :TIMEbase Commands</a>" on page 1036</li> <li>· "<a href="#">:TIMEbase:RANGE</a>" on page 1039</li> <li>· "<a href="#">:TIMEbase:POStion</a>" on page 1038</li> <li>· "<a href="#">:TIMEbase:SCALe</a>" on page 1041</li> <li>· "<a href="#">:TIMEbase:WINDOW:RANGE</a>" on page 1044</li> </ul>



# 33 :TRIGger Commands

Control the trigger modes and parameters for each trigger type. See:

- ["Introduction to :TRIGger Commands" on page 1047](#)
- ["General :TRIGger Commands" on page 1049](#)
- [":TRIGger:DELay Commands" on page 1059](#)
- [":TRIGger:EBURst Commands" on page 1066](#)
- [":TRIGger\[:EDGE\] Commands" on page 1071](#)
- [":TRIGger:GLITch Commands" on page 1077 \(Pulse Width trigger\)](#)
- [":TRIGger:NFC Commands" on page 1086](#)
- [":TRIGger:OR Commands" on page 1096](#)
- [":TRIGger:PATTern Commands" on page 1098](#)
- [":TRIGger:RUNT Commands" on page 1106](#)
- [":TRIGger:SHOLD Commands" on page 1111](#)
- [":TRIGger:TRANsition Commands" on page 1117](#)
- [":TRIGger:TV Commands" on page 1122](#)
- [":TRIGger:USB Commands" on page 1132](#)
- [":TRIGger:ZONE Commands" on page 1137](#)

## Introduction to :TRIGger Commands

The commands in the TRIGger subsystem define the conditions for an internal trigger. Many of these commands are valid in multiple trigger modes.

The default trigger mode is :EDGE.

The trigger subsystem controls the trigger sweep mode and the trigger specification. The trigger sweep (see [":TRIGger:SWEep" on page 1058](#)) can be AUTO or NORMAl.

- **NORMAl** mode – displays a waveform only if a trigger signal is present and the trigger conditions are met. Otherwise the oscilloscope does not trigger and the display is not updated. This mode is useful for low-repetitive-rate signals.
- **AUTO** trigger mode – generates an artificial trigger event if the trigger specification is not satisfied within a preset time, acquires unsynchronized data and displays it.

AUTO mode is useful for signals other than low-repetitive-rate signals. You must use this mode to display a DC signal because there are no edges on which to trigger.

The following trigger types are available (see "[:TRIGger:MODE](#)" on page 1056).

- **Edge triggering**— identifies a trigger by looking for a specified slope and voltage level on a waveform.
- **Nth Edge Burst triggering**— lets you trigger on the Nth edge of a burst that occurs after an idle time.
- **Pulse width triggering**— (:TRIGger:GLITch commands) sets the oscilloscope to trigger on a positive pulse or on a negative pulse of a specified width.
- **Pattern triggering**— identifies a trigger condition by looking for a specified pattern. This pattern is a logical AND combination of the channels. You can also trigger on a specified time duration of a pattern.
- **TV triggering**— is used to capture the complicated waveforms of television equipment. The trigger circuitry detects the vertical and horizontal interval of the waveform and produces triggers based on the TV trigger settings you selected. TV triggering requires greater than  $\frac{1}{4}$  division of sync amplitude with any analog channel as the trigger source.
- **USB (Universal Serial Bus) triggering**— will trigger on a Start of Packet (SOP), End of Packet (EOP), Reset Complete, Enter Suspend, or Exit Suspend signal on the differential USB data lines. USB Low Speed and Full Speed are supported by this trigger.

### Reporting the Setup

Use :TRIGger? to query setup information for the TRIGger subsystem.

### Return Format

The return format for the TRIGger? query varies depending on the current mode. The following is a sample response from the :TRIGger? query. In this case, the query was issued following a \*RST command.

```
:TRIG:MODE EDGE;SWE AUTO;NREJ 0;HFR 0;HOLD +60.000000000000E-09;
:TRIG:EDGE:SOUR CHAN1;LEV +0.00000E+00;SLOP POS;REJ OFF;COUP DC;
:TRIG:ZONE:STAT 0
```

## General :TRIGger Commands

**Table 139** General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:FORCe (see page 1050)	n/a	n/a
:TRIGger:HFReject {{0   OFF}   {1   ON}} (see page 1051)	:TRIGger:HFReject? (see page 1051)	{0   1}
:TRIGger:HOLDoff <holdoff_time> (see page 1052)	:TRIGger:HOLDoff? (see page 1052)	<holdoff_time> ::= 60 ns to 10 s in NR3 format
:TRIGger:LEVel:ASETup (see page 1053)	n/a	n/a
:TRIGger:LEVel:HIGH <level>, <source> (see page 1054)	:TRIGger:LEVel:HIGH? <source> (see page 1054)	<level> ::= .75 x full-scale voltage from center screen in NR3 format. <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:LEVel:LOW <level>, <source> (see page 1055)	:TRIGger:LEVel:LOW? <source> (see page 1055)	<level> ::= .75 x full-scale voltage from center screen in NR3 format. <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:MODE <mode> (see page 1056)	:TRIGger:MODE? (see page 1056)	<mode> ::= {EDGE   GLITCH   PATTern   TV   DELay   EBURst   OR   RUNT   SHOLD   TRANSition   SBUS{1   2}} <return_value> ::= {<mode>   <none>} <none> ::= query returns "NONE" if the :TIMEbase:MODE is ROLL or XY
:TRIGger:NREject {{0   OFF}   {1   ON}} (see page 1057)	:TRIGger:NREject? (see page 1057)	{0   1}
:TRIGger:SWEep <sweep> (see page 1058)	:TRIGger:SWEep? (see page 1058)	<sweep> ::= {AUTO   NORMAl}

## :TRIGger:FORCe

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:FORCe

The :TRIGger:FORCe command causes an acquisition to be captured even though the trigger condition has not been met. This command is equivalent to the front panel **[Force Trigger]** key.

**See Also** • ["Introduction to :TRIGger Commands"](#) on page 1047

**:TRIGger:HFReject****C** (see [page 1334](#))**Command Syntax**    `:TRIGger:HFReject <value>``<value> ::= {{0 | OFF} | {1 | ON}}`

The :TRIGger:HFReject command turns the high frequency reject filter off and on. The high frequency reject filter adds a 50 kHz low-pass filter in the trigger path to remove high frequency components from the trigger waveform. Use this filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.

**Query Syntax**    `:TRIGger:HFReject?`

The :TRIGger:HFReject? query returns the current high frequency reject filter mode.

**Return Format**    `<value><NL>``<value> ::= {0 | 1}`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- "[:TRIGger\[:EDGE\]:REject](#)" on page 1074

**:TRIGger:HOLDoff****C** (see [page 1334](#))**Command Syntax**    `:TRIGger:HOLDoff <holdoff_time>``<holdoff_time> ::= 40 ns to 10 s in NR3 format`

The :TRIGger:HOLDoff command defines the holdoff time value in seconds. Holdoff keeps a trigger from occurring until after a certain amount of time has passed since the last trigger. This feature is valuable when a waveform crosses the trigger level multiple times during one period of the waveform. Without holdoff, the oscilloscope could trigger on each of the crossings, producing a confusing waveform. With holdoff set correctly, the oscilloscope always triggers on the same crossing. The correct holdoff setting is typically slightly less than one period.

**Query Syntax**    `:TRIGger:HOLDoff?`

The :TRIGger:HOLDoff? query returns the holdoff time value for the current trigger mode.

**Return Format**    `<holdoff_time><NL>``<holdoff_time> ::= the holdoff time value in seconds in NR3 format.`**See Also**    · "Introduction to :TRIGger Commands" on page 1047

## :TRIGger:LEVel:ASETUp

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:LEVel:ASETUp

The :TRIGger:LEVel:ASETUp command automatically sets the trigger levels of all displayed analog channels to their waveforms' 50% values.

If AC coupling is used, the trigger levels are set to 0 V.

When High and Low (dual) trigger levels are used (as with Rise/Fall Time and Runt triggers, for example), this command has no effect.

**See Also** • [":TRIGger\[:EDGE\]:LEVel"](#) on page 1073

## :TRIGger:LEVel:HIGH

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:LEVel:HIGH <level>, <source>`

`<level> ::= 0.75 x full-scale voltage from center screen in NR3 format  
for internal triggers`

`<source> ::= CHANnel<n>`

`<n> ::= 1 to (# analog channels) in NR1 format`

The :TRIGger:LEVel:HIGH command sets the high trigger voltage level voltage for the specified source.

High and low trigger levels are used with runt triggers and rise/fall time (transition) triggers.

**Query Syntax**    `:TRIGger:LEVel:HIGH? <source>`

The :TRIGger:LEVel:HIGH? query returns the high trigger voltage level for the specified source.

**Return Format**    `<level><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:LEVel:LOW](#)" on page 1055
- "[:TRIGger:RUNT Commands](#)" on page 1106
- "[:TRIGger:TRANSition Commands](#)" on page 1117
- "[:TRIGger\[:EDGE\]:SOURce](#)" on page 1076

## :TRIGger:LEVel:LOW

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:LEVel:LOW <level>, <source>`

`<level> ::= 0.75 x full-scale voltage from center screen in NR3 format  
for internal triggers`

`<source> ::= CHANnel<n>`

`<n> ::= 1 to (# analog channels) in NR1 format`

The :TRIGger:LEVel:LOW command sets the low trigger voltage level voltage for the specified source.

High and low trigger levels are used with runt triggers and rise/fall time (transition) triggers.

**Query Syntax**    `:TRIGger:LEVel:LOW? <source>`

The :TRIGger:LEVel:LOW? query returns the low trigger voltage level for the specified source.

**Return Format**    `<level><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:LEVel:HIGH](#)" on page 1054
- "[:TRIGger:RUNT Commands](#)" on page 1106
- "[:TRIGger:TRANSition Commands](#)" on page 1117
- "[:TRIGger\[:EDGE\]:SOURce](#)" on page 1076

## :TRIGger:MODE

 (see [page 1334](#))

**Command Syntax**    `:TRIGger:MODE <mode>`

```
<mode> ::= {EDGE | GLITch | PATTern | TV | DELay | EBURst | OR | RUNT
             | SHold | TRANsition | SBUS{1 | 2} | NFC | USB}
```

The :TRIGger:MODE command selects the trigger mode (trigger type).

**Query Syntax**    `:TRIGger:MODE?`

The :TRIGger:MODE? query returns the current trigger mode. If the :TIMEbase:MODE is ROLL or XY, the query returns "NONE".

**Return Format**    `<mode><NL>`

```
<mode> ::= {EDGE | GLIT | PATT | TV | DEL | EBUR | OR | RUNT | SHOL
             | TRAN | SBUS{1 | 2} | NFC | USB}
```

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:SWEep](#)" on page 1058
  - "[:TIMEbase:MODE](#)" on page 1037

**Example Code**

```
' TRIGGER_MODE - Set the trigger mode to EDGE.
myScope.WriteString ":TRIGger:MODE EDGE"
```

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

**:TRIGger:NREJect** (see [page 1334](#))**Command Syntax**    `:TRIGger:NREJect <value>``<value> ::= {{0 | OFF} | {1 | ON}}`

The :TRIGger:NREJect command turns the noise reject filter off and on. When the noise reject filter is on, the trigger circuitry is less sensitive to noise but may require a greater amplitude waveform to trigger the oscilloscope. This command is not valid in TV trigger mode.

**Query Syntax**    `:TRIGger:NREJect?`

The :TRIGger:NREJect? query returns the current noise reject filter mode.

**Return Format**    `<value><NL>``<value> ::= {0 | 1}`**See Also**    · ["Introduction to :TRIGger Commands"](#) on page 1047

## :TRIGger:SWEep

**C** (see [page 1334](#))

**Command Syntax**    `:TRIGger:SWEep <sweep>`

`<sweep> ::= {AUTO | NORMAl}`

The :TRIGger:SWEep command selects the trigger sweep mode.

When AUTO sweep mode is selected, a baseline is displayed in the absence of a signal. If a signal is present but the oscilloscope is not triggered, the unsynchronized signal is displayed instead of a baseline.

When NORMAl sweep mode is selected and no trigger is present, the instrument does not sweep, and the data acquired on the previous trigger remains on the screen.

### NOTE

This feature is called "Mode" on the instrument's front panel.

---

**Query Syntax**    `:TRIGger:SWEep?`

The :TRIGger:SWEep? query returns the current trigger sweep mode.

**Return Format**    `<sweep><NL>`

`<sweep> ::= current trigger sweep mode`

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047

## :TRIGger:DELay Commands

**Table 140** :TRIGger:DELay Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DELay:ARM:SL OPe <slope> (see <a href="#">page 1060</a> )	:TRIGger:DELay:ARM:SL OPe? (see <a href="#">page 1060</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:DELay:ARM:SOURce <source> (see <a href="#">page 1061</a> )	:TRIGger:DELay:ARM:SOURce? (see <a href="#">page 1061</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:DELay:TDELay :TIME <time_value> (see <a href="#">page 1062</a> )	:TRIGger:DELay:TDELay :TIME? (see <a href="#">page 1062</a> )	<time_value> ::= time in seconds in NR3 format
:TRIGger:DELay:TRIGger:COUNT <count> (see <a href="#">page 1063</a> )	:TRIGger:DELay:TRIGger:COUNT? (see <a href="#">page 1063</a> )	<count> ::= integer in NR1 format
:TRIGger:DELay:TRIGger:SLOPe <slope> (see <a href="#">page 1064</a> )	:TRIGger:DELay:TRIGger:SLOPe? (see <a href="#">page 1064</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:DELay:TRIGger:SOURce <source> (see <a href="#">page 1065</a> )	:TRIGger:DELay:TRIGger:SOURce? (see <a href="#">page 1065</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

The :TRIGger:DELay:ARM:SOURce and :TRIGger:DELay:TRIGger:SOURce commands are used to specify the source channel for the arming edge and the trigger edge in the Edge Then Edge trigger.

If an analog channel is selected as a source, the :TRIGger:EDGE:LEVel command is used to set the trigger level.

If a digital channel is selected as the source, the :DIGital<n>:THReShold or :POD<n>:THReShold command is used to set the trigger level.

**:TRIGger:DELay:ARM:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:DELay:ARM:SLOPe <slope>`  
                  `<slope> ::= {NEGative | POSitive}`

The :TRIGger:DELay:ARM:SLOPe command specifies rising (POSitive) or falling (NEGative) for the arming edge in the Edge Then Edge trigger.

**Query Syntax**    `:TRIGger:DELay:ARM:SLOPe?`

The :TRIGger:DELay:ARM:SLOPe? query returns the current arming edge slope setting.

**Return Format**    `<slope><NL>`  
                  `<slope> ::= {NEG | POS}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:DELay:ARM:SOURce" on page 1061
- ":TRIGger:DELay:TDELay:TIME" on page 1062

## :TRIGger:DELay:ARM:SOURce

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:DELay:ARM:SOURce <source>

```
<source> ::= {CHANnel<n> | DIGital<d>}
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :TRIGger:DELay:ARM:SOURce command selects the input used for the arming edge in the Edge Then Edge trigger.

**Query Syntax** :TRIGger:DELay:ARM:SOURce?

The :TRIGger:DELay:ARM:SOURce? query returns the current arming edge source.

**Return Format** <source><NL>

```
<source> ::= {CHAN<n> | DIG<d>}
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:DELay:ARM:SLOPe](#)" on page 1060
- "[:TRIGger:DELay:TDELay:TIME](#)" on page 1062
- "[:TRIGger:MODE](#)" on page 1056

**:TRIGger:DELay:TDELay:TIME****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:DELay:TDELay:TIME <time_value>`  
                      `<time_value> ::= time in seconds in NR3 format`

The :TRIGger:DELay:TDELay:TIME command sets the delay time between the arming edge and the trigger edge in the Edge Then Edge trigger. The time is in seconds and must be from 4 ns to 10 s.

**Query Syntax**    `:TRIGger:DELay:TDELay:TIME?`

The :TRIGger:DELay:TDELay:TIME? query returns current delay time setting.

**Return Format**    `<time value><NL>`  
                      `<time_value> ::= time in seconds in NR3 format`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:DELay:TRIGger:SLOPe" on page 1064
- ":TRIGger:DELay:TRIGger:COUNT" on page 1063

**:TRIGger:DElay:TRIGger:COUNt**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:DElay:TRIGger:COUNt <count>`  
                  `<count> ::= integer in NR1 format`

The :TRIGger:DElay:TRIGger:COUNt command sets the Nth edge of the trigger source to trigger on.

**Query Syntax**    `:TRIGger:DElay:TRIGger:COUNt?`

The :TRIGger:DElay:TRIGger:COUNt? query returns the current Nth trigger edge setting.

**Return Format**    `<count><NL>`  
                  `<count> ::= integer in NR1 format`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:DElay:TRIGger:SLOPe" on page 1064
- ":TRIGger:DElay:TRIGger:SOURce" on page 1065
- ":TRIGger:DElay:TDELay:TIME" on page 1062

**:TRIGger:DELay:TRIGger:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:DELay:TRIGger:SLOPe <slope>`  
`<slope> ::= {NEGative | POSitive}`

The :TRIGger:DELay:TRIGger:SLOPe command specifies rising (POSitive) or falling (NEGative) for the trigger edge in the Edge Then Edge trigger.

**Query Syntax**    `:TRIGger:DELay:TRIGger:SLOPe?`

The :TRIGger:DELay:TRIGger:SLOPe? query returns the current trigger edge slope setting.

**Return Format**    `<slope><NL>`  
`<slope> ::= {NEG | POS}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:DELay:TRIGger:SOURce](#)" on page 1065
- "[:TRIGger:DELay:TDELay:TIME](#)" on page 1062
- "[:TRIGger:DELay:TRIGger:COUNT](#)" on page 1063

## :TRIGger:DELay:TRIGger:SOURce

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:DELay:TRIGger:SOURce <source>

```
<source> ::= {CHANnel<n> | DIGital<d>}
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :TRIGger:DELay:TRIGger:SOURce command selects the input used for the trigger edge in the Edge Then Edge trigger.

**Query Syntax** :TRIGger:DELay:TRIGger:SOURce?

The :TRIGger:DELay:TRIGger:SOURce? query returns the current trigger edge source.

**Return Format** <source><NL>

```
<source> ::= {CHAN<n> | DIG<d>}
```

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:DELay:TRIGger:SLOPe](#)" on page 1064
  - "[:TRIGger:DELay:TDELay:TIME](#)" on page 1062
  - "[:TRIGger:DELay:TRIGger:COUNt](#)" on page 1063
  - "[:TRIGger:MODE](#)" on page 1056

## :TRIGger:EBURst Commands

**Table 141** :TRIGger:EBURst Commands Summary

Command	Query	Options and Query Returns
:TRIGger:EBURst:COUNT <count> (see <a href="#">page 1067</a> )	:TRIGger:EBURst:COUNT? ? (see <a href="#">page 1067</a> )	<count> ::= integer in NR1 format
:TRIGger:EBURst:IDLE <time_value> (see <a href="#">page 1068</a> )	:TRIGger:EBURst:IDLE? ? (see <a href="#">page 1068</a> )	<time_value> ::= time in seconds in NR3 format
:TRIGger:EBURst:SLOPe <slope> (see <a href="#">page 1069</a> )	:TRIGger:EBURst:SLOPe? ? (see <a href="#">page 1069</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:EBURst:SOURce <source> (see <a href="#">page 1070</a> )	:TRIGger:EBURst:SOURce? ? (see <a href="#">page 1070</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

The :TRIGger:EBURst:SOURce command is used to specify the source channel for the Nth Edge Burst trigger. If an analog channel is selected as the source, the :TRIGger:EDGE:LEVel command is used to set the Nth Edge Burst trigger level. If a digital channel is selected as the source, the :DIGital<n>:THreshold or :POD<n>:THreshold command is used to set the Nth Edge Burst trigger level.

**:TRIGger:EBURst:COUNt****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:EBURst:COUNt <count>`  
                  `<count> ::= integer in NR1 format`

The :TRIGger:EBURst:COUNt command sets the Nth edge at burst counter resource. The edge counter is used in the trigger stage to determine which edge in a burst will generate a trigger.

**Query Syntax**    `:TRIGger:EBURst:COUNt?`

The :TRIGger:EBURst:COUNt? query returns the current Nth edge of burst edge counter setting.

**Return Format**    `<count><NL>`  
                  `<count> ::= integer in NR1 format`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:EBURst:SLOPe" on page 1069
- ":TRIGger:EBURst:IDLE" on page 1068

**:TRIGger:EBURst:IDLE****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:EBURst:IDLE <time_value>`    `<time_value> ::= time in seconds in NR3 format`

The :TRIGger:EBURst:IDLE command sets the Nth edge in a burst idle resource in seconds from 10 ns to 10 s. The timer is used to set the minimum time before the next burst.

**Query Syntax**    `:TRIGger:EBURst:IDLE?`

The :TRIGger:EBURst:IDLE? query returns current Nth edge in a burst idle setting.

**Return Format**    `<time value><NL>`    `<time_value> ::= time in seconds in NR3 format`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:EBURst:SLOPe" on page 1069
- ":TRIGger:EBURst:COUNT" on page 1067

**:TRIGger:EBURst:SLOPe****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:EBURst:SLOPe <slope>`    `<slope> ::= {NEGative | POSitive}`

The :TRIGger:EBURst:SLOPe command specifies whether the rising edge (POSitive) or falling edge (NEGative) of the Nth edge in a burst will generate a trigger.

**Query Syntax**    `:TRIGger:EBURst:SLOPe?`

The :TRIGger:EBURst:SLOPe? query returns the current Nth edge in a burst slope.

**Return Format**    `<slope><NL>`    `<slope> ::= {NEG | POS}`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:EBURst:IDLE" on page 1068
- ":TRIGger:EBURst:COUNT" on page 1067

**:TRIGger:EBURst:SOURce**(see [page 1334](#))**Command Syntax**    `:TRIGger:EBURst:SOURce <source>`    `<source> ::= {CHANnel<n> | DIGital<d>}`    `<n> ::= 1 to (# analog channels) in NR1 format`    `<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :TRIGger:EBURst:SOURce command selects the input that produces the Nth edge burst trigger.

**Query Syntax**    `:TRIGger:EBURst:SOURce?`

The :TRIGger:EBURst:SOURce? query returns the current Nth edge burst trigger source. If all channels are off, the query returns "NONE."

**Return Format**    `<source><NL>`    `<source> ::= {CHAN<n> | DIG<d>}`**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":TRIGger:MODE"](#) on page 1056

## :TRIGger[:EDGE] Commands

**Table 142**:TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE] :COUPLing {AC   DC   LFReject} (see page 1072)	:TRIGger[:EDGE] :COUPLing? (see page 1072)	{AC   DC   LFReject}
:TRIGger[:EDGE] :LEVel <level> [,<source>] (see page 1073)	:TRIGger[:EDGE] :LEVel? [<source>] (see page 1073)	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers, <level> ::= ±(external range setting) in NR3 format. For digital channels (MSO models), <level> ::= ±8 V. <source> ::= {CHANnel<n>   EXTERNAL} for DSO models <source> ::= {CHANnel<n>   DIGItal<d>   EXTERNAL } for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger[:EDGE] :REJect {OFF   LFReject   HFReject} (see page 1074)	:TRIGger[:EDGE] :REJect? (see page 1074)	{OFF   LFReject   HFReject}
:TRIGger[:EDGE] :SLOPe <polarity> (see page 1075)	:TRIGger[:EDGE] :SLOPe? (see page 1075)	<polarity> ::= {POSitive   NEGative   EITHer   ALTernate}
:TRIGger[:EDGE] :SOURce <source> (see page 1076)	:TRIGger[:EDGE] :SOURce? (see page 1076)	<source> ::= {CHANnel<n>   EXTERNAL   LINE   WGEN   WGEN1   WMOD} for the DSO models <source> ::= {CHANnel<n>   DIGItal<d>   EXTERNAL   LINE   WGEN   WGEN1   WMOD} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format Note: WGEN and WGEN1 are equivalent.

## :TRIGger[:EDGE]:COUpling



(see [page 1334](#))

**Command Syntax**    `:TRIGger[:EDGE] :COUpling <coupling>`  
`<coupling> ::= {AC | DC | LFReject}`

The :TRIGger[:EDGE]:COUpling command sets the input coupling for the selected trigger sources. The coupling can be set to AC, DC, or LFReject.

- AC coupling places a high-pass filter (10 Hz for analog channels, and 3.5 Hz for all External trigger inputs) in the trigger path, removing dc offset voltage from the trigger waveform. Use AC coupling to get a stable edge trigger when your waveform has a large dc offset.
- LFReject coupling places a 50 KHz high-pass filter in the trigger path.
- DC coupling allows dc and ac signals into the trigger path.

### NOTE

The :TRIGger[:EDGE]:COUpling and the :TRIGger[:EDGE]:REJect selections are coupled. Changing the setting of the :TRIGger[:EDGE]:REJect can change the COUpling setting.

**Query Syntax**    `:TRIGger[:EDGE] :COUpling?`

The :TRIGger[:EDGE]:COUpling? query returns the current coupling selection.

**Return Format**    `<coupling><NL>`  
`<coupling> ::= {AC | DC | LFR}`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger\[:EDGE\]:REJect](#)" on page 1074

## :TRIGger[:EDGE]:LEVel

**C** (see [page 1334](#))

**Command Syntax**

```
:TRIGger[:EDGE]:LEVel <level>
<level> ::= <level>[,<source>]
<level> ::= 0.75 x full-scale voltage from center screen in NR3 format
           for internal triggers
<level> ::= ±(external range setting) in NR3 format
           for external triggers
<level> ::= ±8 V for digital channels (MSO models)
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGital<d> | EXTERNAL}
           for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :TRIGger[:EDGE]:LEVel command sets the trigger level voltage for the active trigger source.

### NOTE

If the optional source is specified and is not the active source, the level on the active source is not affected and the active source is not changed.

**Query Syntax**

```
:TRIGger[:EDGE]:LEVel? [<source>]
```

The :TRIGger[:EDGE]:LEVel? query returns the trigger level of the current trigger source.

**Return Format**

```
<level><NL>
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger\[:EDGE\]:SOURce](#)" on page 1076
- "[:EXTERNAL:RANGE](#)" on page 356
- "[:POD<n>:THreshold](#)" on page 605
- "[:DIGITAL<d>:THreshold](#)" on page 327

## :TRIGger[:EDGE]:REJect



(see [page 1334](#))

**Command Syntax**    `:TRIGger[:EDGE]:REJect <reject>`

`<reject> ::= {OFF | LFRReject | HFReject}`

The :TRIGger[:EDGE]:REJect command turns the low-frequency or high-frequency reject filter on or off. You can turn on one of these filters at a time.

- The high frequency reject filter adds a 50 kHz low-pass filter in the trigger path to remove high frequency components from the trigger waveform. Use the high frequency reject filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- The low frequency reject filter adds a 50 kHz high-pass filter in series with the trigger waveform to remove any unwanted low frequency components from a trigger waveform, such as power line frequencies, that can interfere with proper triggering.

### NOTE

The :TRIGger[:EDGE]:REJect and the :TRIGger[:EDGE]:COUPLing selections are coupled. Changing the setting of the :TRIGger[:EDGE]:COUPLing can change the COUPLing setting.

**Query Syntax**    `:TRIGger[:EDGE]:REJect?`

The :TRIGger[:EDGE]:REJect? query returns the current status of the reject filter.

**Return Format**    `<reject><NL>`

`<reject> ::= {OFF | LFR | HFR}`

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":TRIGger:HFReject](#)" on page 1051
  - "[":TRIGger\[:EDGE\]:COUPLing](#)" on page 1072

## :TRIGger[:EDGE]:SLOPe

**C** (see [page 1334](#))

**Command Syntax** :TRIGger[:EDGE]:SLOPe <slope>

<slope> ::= {NEGative | POSitive | EITHer | ALTernate}

The :TRIGger[:EDGE]:SLOPe command specifies the slope of the edge for the trigger. The SLOPe command is not valid in TV trigger mode. Instead, use :TRIGger:TV:POLarity to set the polarity in TV trigger mode.

**Query Syntax** :TRIGger[:EDGE]:SLOPe?

The :TRIGger[:EDGE]:SLOPe? query returns the current trigger slope.

**Return Format** <slope><NL>

<slope> ::= {NEG | POS | EITH | ALT}

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:MODE](#)" on page 1056
  - "[:TRIGger:TV:POLarity](#)" on page 1125

**Example Code**

```
' TRIGGER_EDGE_SLOPE - Sets the slope of the edge for the trigger.  
  
' Set the slope to positive.  
myScope.WriteString ":TRIGGER:EDGE:SLOPE POSITIVE"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :TRIGger[:EDGE]:SOURce

 (see [page 1334](#))

<b>Command Syntax</b>	<code>:TRIGger[:EDGE]:SOURce &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   EXTERNAL   LINE   WGEN   WGEN1   WMOD} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGITAL&lt;d&gt;   EXTERNAL   LINE   WGEN   WGEN1   WMOD} for the MSO models</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	<b>Note:</b> WAVE and WGEN1 are equivalent.

The :TRIGger[:EDGE]:SOURce command selects the input that produces the trigger.

- EXTERNAL – triggers on the rear panel EXT TRIG IN signal.
- LINE – triggers at the 50% level of the rising or falling edge of the AC power source signal.
- WGEN, WGEN1 – triggers at the 50% level of the rising edge of the waveform generator output signal. This option is not available when the DC, NOISe, or CARDiac waveforms are selected.
- WMOD – when waveform generator FSK or FM modulation is used, triggers at the 50% level of the rising edge of the modulating signal.

<b>Query Syntax</b>	<code>:TRIGger[:EDGE]:SOURce?</code>
---------------------	--------------------------------------

The :TRIGger[:EDGE]:SOURce? query returns the current source. If all channels are off, the query returns "NONE."

<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
	<code>&lt;source&gt; ::= {CHAN&lt;n&gt;   EXT   LINE   WGEN   WGEN1   WMOD   NONE} for the DSO models</code>
	<code>&lt;source&gt; ::= {CHAN&lt;n&gt;   DIG&lt;d&gt;   EXTERNAL   LINE   WGEN   WGEN1   WMOD   NONE} for the MSO models</code>

<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>• "<a href="#">:TRIGger:MODE</a>" on page 1056</li> </ul>
-----------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<b>Example Code</b>	<pre>' TRIGGER_EDGE_SOURCE - Selects the channel that actually produces the ' edge trigger. Any channel can be selected. myScope.WriteString ":TRIGger:EDGE:SOURce CHANnel1"</pre>
---------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

See complete example programs at: [Chapter 42, “Programming Examples,”](#) starting on page 1343

## :TRIGger:GLITch Commands

**Table 143:** :TRIGger:GLITch Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITch:GREAt erthan <greater_than_time>[s uffix] (see <a href="#">page 1079</a> )	:TRIGger:GLITch:GREAt erthan? (see <a href="#">page 1079</a> )	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:LESSt han <less_than_time>[suff ix] (see <a href="#">page 1080</a> )	:TRIGger:GLITch:LESSt han? (see <a href="#">page 1080</a> )	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:LEVel <level> [<source>] (see <a href="#">page 1081</a> )	:TRIGger:GLITch:LEVel ? (see <a href="#">page 1081</a> )	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers (DSO models), <level> ::= ±(external range setting) in NR3 format. For digital channels (MSO models), <level> ::= ±8 V. <source> ::= {CHANnel<n>   EXTERNAL} for DSO models <source> ::= {CHANnel<n>   DIGITAL<d>} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:GLITch:POLAr ity <polarity> (see <a href="#">page 1082</a> )	:TRIGger:GLITch:POLAr ity? (see <a href="#">page 1082</a> )	<polarity> ::= {POSitive   NEGative}
:TRIGger:GLITch:QUALi fier <qualifier> (see <a href="#">page 1083</a> )	:TRIGger:GLITch:QUALi fier? (see <a href="#">page 1083</a> )	<qualifier> ::= {GREaterthan   LESSthan   RANGE}

**Table 143:**:TRIGger:GLITch Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:GLITch:RANGE <less_than_time>[suffix], <greater_than_time>[suffix] (see <a href="#">page 1084</a> )	:TRIGger:GLITch:RANGE? ? (see <a href="#">page 1084</a> )	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:GLITch:SOURce <source> (see <a href="#">page 1085</a> )	:TRIGger:GLITch:SOURce? ? (see <a href="#">page 1085</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

## :TRIGger:GLITch:GREaterthan

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:GLITch:GREaterthan <greater_than_time>[<suffix>]`

`<greater_than_time> ::= floating-point number in NR3 format`

`<suffix> ::= {s | ms | us | ns | ps}`

The :TRIGger:GLITch:GREaterthan command sets the minimum pulse width duration for the selected :TRIGger:GLITch:SOURce.

**Query Syntax**    `:TRIGger:GLITch:GREaterthan?`

The :TRIGger:GLITch:GREaterthan? query returns the minimum pulse width duration time for :TRIGger:GLITch:SOURce.

**Return Format**    `<greater_than_time><NL>`

`<greater_than_time> ::= floating-point number in NR3 format.`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:GLITch:SOURce" on page 1085
- ":TRIGger:GLITch:QUALifier" on page 1083
- ":TRIGger:MODE" on page 1056

**:TRIGger:GLITch:LESSthan**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:GLITch:LESSthan <less_than_time>[<suffix>]`  
`<less_than_time> ::= floating-point number in NR3 format`  
`<suffix> ::= {s | ms | us | ns | ps}`

The :TRIGger:GLITch:LESSthan command sets the maximum pulse width duration for the selected :TRIGger:GLITch:SOURce.

**Query Syntax**    `:TRIGger:GLITch:LESSthan?`

The :TRIGger:GLITch:LESSthan? query returns the pulse width duration time for :TRIGger:GLITch:SOURce.

**Return Format**    `<less_than_time><NL>`  
`<less_than_time> ::= floating-point number in NR3 format.`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:GLITch:SOURce" on page 1085
- ":TRIGger:GLITch:QUALifier" on page 1083
- ":TRIGger:MODE" on page 1056

## :TRIGger:GLITch:LEVel

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:GLITch:LEVel <level_argument>`

`<level_argument> ::= <level>[, <source>]`

`<level> ::= .75 x full-scale voltage from center screen in NR3 format  
for internal triggers`

`<level> ::= ±(external range setting) in NR3 format  
for external triggers (DSO models)`

`<level> ::= ±8 V for digital channels (MSO models)`

`<source> ::= {CHANnel<n> | EXTERNAL} for DSO models`

`<source> ::= {CHANnel<n> | DIGital<d>} for MSO models`

`<n> ::= 1 to (# analog channels) in NR1 format`

`<d> ::= 0 to (# digital channels - 1) in NR1 format`

The :TRIGger:GLITch:LEVel command sets the trigger level voltage for the active pulse width trigger.

**Query Syntax**    `:TRIGger:GLITch:LEVel?`

The :TRIGger:GLITch:LEVel? query returns the trigger level of the current pulse width trigger mode. If all channels are off, the query returns "NONE."

**Return Format**    `<level_argument><NL>`

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":TRIGger:MODE"](#) on page 1056
  - "[":TRIGger:GLITch:SOURce"](#) on page 1085
  - "[":EXTERNAL:RANGE"](#) on page 356

**:TRIGger:GLITch:POLarity**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:GLITch:POLarity <polarity>`  
`<polarity> ::= {POSitive | NEGative}`

The :TRIGger:GLITch:POLarity command sets the polarity for the glitch pulse width trigger.

**Query Syntax**    `:TRIGger:GLITch:POLarity?`

The :TRIGger:GLITch:POLarity? query returns the glitch pulse width trigger polarity.

**Return Format**    `<polarity><NL>`  
`<polarity> ::= {POS | NEG}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":TRIGger:GLITch:SOURce" on page 1085

**:TRIGger:GLITch:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:GLITch:QUALifier <operator>``<operator> ::= {GREaterthan | LESSthan | RANGE}`

This command sets the mode of operation of the glitch pulse width trigger. The oscilloscope can trigger on a pulse width that is greater than a time value, less than a time value, or within a range of time values.

**Query Syntax**    `:TRIGger:GLITch:QUALifier?`

The `:TRIGger:GLITch:QUALifier?` query returns the glitch pulse width qualifier.

**Return Format**    `<operator><NL>``<operator> ::= {GRE | LESS | RANG}`**See Also**

- "Introduction to [:TRIGger Commands](#)" on page 1047
- "[:TRIGger:GLITch:SOURce](#)" on page 1085
- "[:TRIGger:MODE](#)" on page 1056

## :TRIGger:GLITch:RANGE

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:GLITch:RANGE <less_than_time>[suffix], <greater_than_time>[suffix]`

`<less_than_time>` ::= (15 ns - 10 seconds) in NR3 format

`<greater_than_time>` ::= (10 ns - 9.99 seconds) in NR3 format

`[suffix]` ::= {s | ms | us | ns | ps}

The :TRIGger:GLITch:RANGE command sets the pulse width duration for the selected :TRIGger:GLITch:SOURce. You can enter the parameters in any order – the smaller value becomes the <greater\_than\_time> and the larger value becomes the <less\_than\_time>.

**Query Syntax**    `:TRIGger:GLITch:RANGE?`

The :TRIGger:GLITch:RANGE? query returns the pulse width duration time for :TRIGger:GLITch:SOURce.

**Return Format**    `<less_than_time>, <greater_than_time><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:GLITch:SOURce](#)" on page 1085
- "[":TRIGger:GLITch:QUALifier](#)" on page 1083
- "[":TRIGger:MODE](#)" on page 1056

## :TRIGger:GLITch:SOURce

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:GLITch:SOURce <source>

```
<source> ::= {DIGItal<d> | CHANnel<n>}  
<n> ::= 1 to (# analog channels) in NR1 format  
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :TRIGger:GLITch:SOURce command selects the channel that produces the pulse width trigger.

**Query Syntax** :TRIGger:GLITch:SOURce?

The :TRIGger:GLITch:SOURce? query returns the current pulse width source. If all channels are off, the query returns "NONE".

**Return Format** <source><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:MODE](#)" on page 1056
  - "[:TRIGger:GLITch:LEVel](#)" on page 1081
  - "[:TRIGger:GLITch:POLarity](#)" on page 1082
  - "[:TRIGger:GLITch:QUALifier](#)" on page 1083
  - "[:TRIGger:GLITch:RANGE](#)" on page 1084

**Example Code**

- "[Example Code](#)" on page 1076

## :TRIGger:NFC Commands

NFC (Near Field Communication) triggering is used to capture waveforms used in NFC testing. The NFC trigger mode is license-enabled.

**Table 144:** :TRIGger:NFC Commands Summary

Command	Query	Options and Query Returns
:TRIGger:NFC:AEVENT <arm_event> (see <a href="#">page 1087</a> )	:TRIGger:NFC:AEVENT? (see <a href="#">page 1087</a> )	<arm_event> ::= {NONE   ASReq   AALLreq   AEITher   BSReq   BALLreq   BEITher   FSReq}
n/a	:TRIGger:NFC:ATTIME? (see <a href="#">page 1088</a> )	<time> ::= seconds in NR3 format
:TRIGger:NFC:SOURCE <source> (see <a href="#">page 1089</a> )	:TRIGger:NFC:SOURce? (see <a href="#">page 1089</a> )	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:NFC:STANDARD <standard> (see <a href="#">page 1090</a> )	:TRIGger:NFC:STANDARD? (see <a href="#">page 1090</a> )	<standard> ::= {{A   A106}   {B   B106}   F212   F424}
:TRIGger:NFC:TEVENT <trigger_event> (see <a href="#">page 1091</a> )	:TRIGger:NFC:TEVENT? (see <a href="#">page 1091</a> )	<trigger_event> ::= {ATRigger   ASReq   AALLreq   AEITher   ASDDreq   BSReq   BALLreq   BEITher   BATTRib   FSReq   FAReq   FPRreamble}
n/a	:TRIGger:NFC:TIMeout? (see <a href="#">page 1093</a> )	{0   1}
:TRIGger:NFC:TIMeout: ENABLE {{0   OFF}   {1   ON}} (see <a href="#">page 1094</a> )	:TRIGger:NFC:TIMeout: ENABLE? (see <a href="#">page 1094</a> )	{0   1}
:TRIGger:NFC:TIMeout: TIME <time> (see <a href="#">page 1095</a> )	:TRIGger:NFC:TIMeout: TIME? (see <a href="#">page 1095</a> )	<time> ::= seconds in NR3 format

## :TRIGger:NFC:AEVENT

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:NFC:AEVENT <arm\_event>

```
<arm_event> ::= {NONE | ASReq | AALLreq | AEITher | BSReq | BALLreq  
| BEITher | FSReq}
```

When the ATRigger (Arm & Trigger) trigger event is selected (by :TRIGger:NFC:TEVENT), the :TRIGger:NFC:AEVENT command specifies the arm event. Valid arm event settings depend on the signaling technology selected (by :TRIGger:NFC:STANDARD):

Signaling Technology	Arm Event	Description
NFC-A	ASReq	SENS_REQ
	AALLreq	ALL_REQ
	AEITher	Either the SENS_REQ or ALL_REQ events will arm.
NFC-B	BSReq	SENSB_REQ
	BALLreq	ALLB_REQ
	BEITher	Either the SENSB_REQ or ALLB_REQ events will arm.
NFC-F	FSReq	SENSF_REQ

When a trigger event other than ATRigger (Arm & Trigger) is selected, the arm event is NONE.

**Query Syntax** :TRIGger:NFC:AEVENT?

The :TRIGger:NFC:AEVENT? query returns the specified arm event.

**Return Format** <arm\_event><NL>

```
<arm_event> ::= {NONE | ASR | AALL | AEIT | BSR | BALL | BEIT | FSR}
```

- See Also**
- "[:TRIGger:NFC:ATTIME](#)" on page 1088
  - "[:TRIGger:NFC:SOURce](#)" on page 1089
  - "[:TRIGger:NFC:STANDARD](#)" on page 1090
  - "[:TRIGger:NFC:TEVENT](#)" on page 1091
  - "[:TRIGger:NFC:TIMEOUT](#)" on page 1093
  - "[:TRIGger:NFC:TIMEOUT:ENABLE](#)" on page 1094
  - "[:TRIGger:NFC:TIMEOUT:TIME](#)" on page 1095

**:TRIGger:NFC:ATTime****N** (see [page 1334](#))**Query Syntax** `:TRIGger:NFC:ATTime?`

The `:TRIGger:NFC:ATTime?` query returns the time between the arm and the trigger when the ATRigger (Arm & Trigger) trigger event is selected.

**Return Format** `<time><NL>`

`<time>` ::= seconds in NR3 format

**See Also**

- [":TRIGger:NFC:AEVENT"](#) on page 1087
- [":TRIGger:NFC:SOURce"](#) on page 1089
- [":TRIGger:NFC:STANDARD"](#) on page 1090
- [":TRIGger:NFC:TEVENT"](#) on page 1091
- [":TRIGger:NFC:TIMEOUT"](#) on page 1093
- [":TRIGger:NFC:TIMEOUT:ENABLE"](#) on page 1094
- [":TRIGger:NFC:TIMEOUT:TIME"](#) on page 1095

## :TRIGger:NFC:SOURce

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:NFC:SOURce <source>

```
<source> ::= CHANnel<n>
<n> ::= 1 to (# analog channels) in NR1 format
```

The :TRIGger:NFC:SOURce command selects the input waveform source for the NFC trigger. You can choose an analog input channel.

**Query Syntax** :TRIGger:NFC:SOURce?

The :TRIGger:NFC:SOURce? query returns the input waveform source setting.

**Return Format** <source><NL>

```
<source> ::= CHAN<n>
```

- See Also**
- "[:TRIGger:NFC:AEvent](#)" on page 1087
  - "[:TRIGger:NFC:ATTime](#)" on page 1088
  - "[:TRIGger:NFC:STANDARD](#)" on page 1090
  - "[:TRIGger:NFC:TEVENT](#)" on page 1091
  - "[:TRIGger:NFC:TIMEOUT](#)" on page 1093
  - "[:TRIGger:NFC:TIMEOUT:ENABLE](#)" on page 1094
  - "[:TRIGger:NFC:TIMEOUT:TIME](#)" on page 1095

## :TRIGger:NFC:STANDARD

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:NFC:STANDARD <standard>`

```
<standard> ::= {{A | A106} | {B | B106} | F212 | F424}
```

The :TRIGger:NFC:STANDARD command selects the signaling technology used by the input signal:

- A or A106 – NFC-A standard, 106 kbits/s.
- B or B106 – NFC-A standard, 106 kbits/s.
- F212 – NFC-F standard, 212 kbits/s.
- F424 – NFC-F standard, 424 kbits/s.

**Query Syntax**    `:TRIGger:NFC:STANDARD?`

The :TRIGger:NFC:STANDARD? query returns the signaling technology setting.

**Return Format**    `<standard><NL>`

```
<standard> ::= {{A | A106} | {B | B106} | F212 | F424}
```

**See Also**    [":TRIGger:NFC:AEvent"](#) on page 1087

- [":TRIGger:NFC:ATTime"](#) on page 1088
- [":TRIGger:NFC:SOURce"](#) on page 1089
- [":TRIGger:NFC:TEVent"](#) on page 1091
- [":TRIGger:NFC:TIMeout"](#) on page 1093
- [":TRIGger:NFC:TIMeout:ENABLE"](#) on page 1094
- [":TRIGger:NFC:TIMeout:TIME"](#) on page 1095

## :TRIGger:NFC:TEVent

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:NFC:TEVent <trigger\_event>

```
<trigger_event> ::= {ATRigger | ASReq | AALLreq | AEITher | ASDDreq
| BSReq | BALLreq | BEITher | BATTrib | FSReq | FAReq | FPRreamble}
```

The :TRIGger:NFC:TEVent command specifies the trigger event. Valid trigger event settings depend on the signaling technology selected (by :TRIGger:NFC:STANDARD):

Signaling Technology	Trigger Event	Description
NFC-A	ATRigger	The arm event is specified by :TRIGger:NFC:AEvent, and the trigger event is SDD_REQ.
	ASReq	SENS_REQ
	AALLreq	ALL_REQ
	AEITher	Either the SENS_REQ or ALL_REQ events will cause a trigger.
	ASDDreq	SDD_REQ
NFC-B	ATRigger	The arm event is specified by :TRIGger:NFC:AEvent, and the trigger event is ATTRIB.
	BSReq	SENSB_REQ
	BALLreq	ALLB_REQ
	BEITher	Either the SENSB_REQ or ALLB_REQ events will cause a trigger.
	BATTrib	ATTRIB
NFC-F	ATRigger	The arm event is specified by :TRIGger:NFC:AEvent, and the trigger event is ATR_REQ.
	FSReq	SENSF_REQ
	FAReq	ATR_REQ
	FPRreamble	The preamble sequence that begins a data frame will cause a trigger.

**Query Syntax** :TRIGger:NFC:TEVent?

The :TRIGger:NFC:TEVent? query returns the specified trigger event.

**Return Format** <trigger\_event><NL>

```
<trigger_event> ::= {ATR | ASR | AALL | AEIT | ASDD | BSR | BALL
| BEIT | BATT | FSR | FAR | FPR}
```

**See Also** • [":TRIGger:NFC:AEvent"](#) on page 1087

- "[:TRIGger:NFC:ATTime](#)" on page 1088
- "[:TRIGger:NFC:SOURce](#)" on page 1089
- "[:TRIGger:NFC:STANDARD](#)" on page 1090
- "[:TRIGger:NFC:TIMEout](#)" on page 1093
- "[:TRIGger:NFC:TIMEout:ENABLE](#)" on page 1094
- "[:TRIGger:NFC:TIMEout:TIME](#)" on page 1095

## :TRIGger:NFC:TIMEout

**N** (see [page 1334](#))

**Query Syntax** :TRIGger:NFC:TIMEout?

The :TRIGger:NFC:TIMEout? query returns whether the timeout occurred.

With the ATRigger (Arm & Trigger) trigger event, if the second event does not occur within the timeout period, the oscilloscope triggers when the timeout period expires.

A return value of 1 says the desired trigger event did not occur within the specified time after the arm event.

A return value of 0 says the desired trigger event occurred before the timeout.

**Return Format** <timeout\_occurred><NL>

{0 | 1}

**See Also**

- [":TRIGger:NFC:AEVENT"](#) on page 1087
- [":TRIGger:NFC:ATTIME"](#) on page 1088
- [":TRIGger:NFC:SOURce"](#) on page 1089
- [":TRIGger:NFC:STANDARD"](#) on page 1090
- [":TRIGger:NFC:TEVENT"](#) on page 1091
- [":TRIGger:NFC:TIMEOUT:ENABLE"](#) on page 1094
- [":TRIGger:NFC:TIMEOUT:TIME"](#) on page 1095

**:TRIGger:NFC:TIMEout:ENABLE**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:NFC:TIMEout:ENABLE {1 | ON}`

The :TRIGger:NFC:TIMEout:ENABLE command enables the timeout period. Currently, ON is the only valid setting.

With the ATRigger (Arm & Trigger) trigger event, if the second event does not occur within the timeout period, the oscilloscope triggers when the timeout period expires.

**Query Syntax**    `:TRIGger:NFC:TIMEout:ENABLE?`

The :TRIGger:NFC:TIMEout:ENABLE? query returns the timeout period enable setting.

**Return Format**    `<seting><NL>`

`{1}`

- See Also**
- [":TRIGger:NFC:AEvent" on page 1087](#)
  - [":TRIGger:NFC:ATTime" on page 1088](#)
  - [":TRIGger:NFC:SOURce" on page 1089](#)
  - [":TRIGger:NFC:STANDARD" on page 1090](#)
  - [":TRIGger:NFC:TEVENT" on page 1091](#)
  - [":TRIGger:NFC:TIMEOUT" on page 1093](#)
  - [":TRIGger:NFC:TIMEOUT:TIME" on page 1095](#)

**:TRIGger:NFC:TIMEout:TIME****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:NFC:TIMEout:TIME <time>`  
                  `<time> ::= seconds in NR3 format`

The :TRIGger:NFC:TIMEout:TIME command specifies the timeout period. With the ATRigger (Arm & Trigger) trigger event, if the second event does not occur within the timeout period, the oscilloscope triggers when the timeout period expires.

**Query Syntax**    `:TRIGger:NFC:TIMEout:TIME?`

The :TRIGger:NFC:TIMEout:TIME? query returns the specified timeout period.

**Return Format**    `<time><NL>`  
                  `<time> ::= seconds in NR3 format`

**See Also**

- "[:TRIGger:NFC:AEvent](#)" on page 1087
- "[:TRIGger:NFC:ATTime](#)" on page 1088
- "[:TRIGger:NFC:SOURce](#)" on page 1089
- "[:TRIGger:NFC:STANDARD](#)" on page 1090
- "[:TRIGger:NFC:TEVENT](#)" on page 1091
- "[:TRIGger:NFC:TIMEOUT](#)" on page 1093
- "[:TRIGger:NFC:TIMEOUT:ENABLE](#)" on page 1094

## :TRIGger:OR Commands

**Table 145:**TRIGger:OR Commands Summary

Command	Query	Options and Query Returns
:TRIGger:OR <string> (see <a href="#">page 1097</a> )	:TRIGger:OR? (see <a href="#">page 1097</a> )	<p>&lt;string&gt; ::= "nn...n" where n ::= {R   F   E   X}</p> <p>R = rising edge, F = falling edge, E = either edge, X = don't care.</p> <p>Each character in the string is for an analog or digital channel as shown on the front panel display.</p>

## :TRIGger:OR

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:OR <string>

<string> ::= "nn...n" where n ::= {R | F | E | X}

R = rising edge, F = falling edge, E = either edge, X = don't care.

The :TRIGger:OR command specifies the edges to include in the OR'ed edge trigger.

In the <string> parameter, each bit corresponds to a channel as described in the following table:

Oscilloscope Models	Value and Mask Bit Assignments
<b>4 analog + 16 digital channels (mixed-signal)</b>	Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 4 through 1.
<b>2 analog + 16 digital channels (mixed-signal)</b>	Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - analog channels 2 and 1.
<b>4 analog channels only</b>	Bits 0 through 3 - analog channels 4 through 1.
<b>2 analog channels only</b>	Bits 0 and 1 - analog channels 2 and 1.

**Query Syntax** :TRIGger:OR?

The :TRIGger:OR? query returns the current OR'ed edge trigger string.

**Return Format** <string><NL>

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1047
  - [":TRIGger:MODE"](#) on page 1056

## :TRIGger:PATTERn Commands

**Table 146:**TRIGger:PATTERn Commands Summary

Command	Query	Options and Query Returns
:TRIGger:PATTERn<string>[,<edge_source>,<edge>] (see page 1099)	:TRIGger:PATTERn? (see <a href="#">page 1100</a> )	<string> ::= "nn...n" where n ::= {0   1   X   R   F} when <base> = ASCII <string> ::= "0xnn...n" where n ::= {0,...,9   A,...,F   X   \$} when <base> = HEX <edge_source> ::= {CHANnel<n>   NONE} for DSO models <edge_source> ::= {CHANnel<n>   DIGital<d>   NONE} for MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format <edge> ::= {POSitive   NEGative}
:TRIGger:PATTERn:FORM at <base> (see page 1101)	:TRIGger:PATTERn:FORM at? (see <a href="#">page 1101</a> )	<base> ::= {ASCII   HEX}
:TRIGger:PATTERn:GREaterthan<greater_than_time>[suffix] (see <a href="#">page 1102</a> )	:TRIGger:PATTERn:GREaterthan? (see <a href="#">page 1102</a> )	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:PATTERn:LESS than<less_than_time>[suffix] (see <a href="#">page 1103</a> )	:TRIGger:PATTERn:LESS than? (see <a href="#">page 1103</a> )	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:PATTERn:QUALifier <qualifier> (see <a href="#">page 1104</a> )	:TRIGger:PATTERn:QUALifier? (see <a href="#">page 1104</a> )	<qualifier> ::= {ENTERed   GREaterthan   LESSthan   INRange   OUTRange   TIMeout}
:TRIGger:PATTERn:RANGE<less_than_time>[suffix],<greater_than_time>[suffix] (see <a href="#">page 1105</a> )	:TRIGger:PATTERn:RANGE? (see <a href="#">page 1105</a> )	<less_than_time> ::= 15 ns to 10 seconds in NR3 format <greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format [suffix] ::= {s   ms   us   ns   ps}

## :TRIGger:PATTERn

**C** (see [page 1334](#))

**Command Syntax** :TRIGger:PATTERn <pattern>

```
<pattern> ::= <string>[,<edge_source>,<edge>]
<string> ::= "nn...n" where n ::= {0 | 1 | X | R | F} when
            <base> = ASCII
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X | $} when
            <base> = HEX
<edge_source> ::= {CHANnel<n> | NONE} for DSO models
<edge_source> ::= {CHANnel<n> | DIGItal<d>
                  | NONE} for MSO models
<n> ::= 1 to (# of analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
<edge> ::= {POSitive | NEGative}
```

The :TRIGger:PATTERn command specifies the channel values to be used in the pattern trigger.

In the <string> parameter, each bit corresponds to a channel as described in the following table:

Oscilloscope Models	Value and Mask Bit Assignments
<b>4 analog + 16 digital channels (mixed-signal)</b>	Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 4 through 1.
<b>2 analog + 16 digital channels (mixed-signal)</b>	Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - analog channels 2 and 1.
<b>4 analog channels only</b>	Bits 0 through 3 - analog channels 4 through 1.
<b>2 analog channels only</b>	Bits 0 and 1 - analog channels 2 and 1.

The format of the <string> parameter depends on the :TRIGger:PATTERn:FORMat command setting:

- When the format is ASCII, the string looks just like the string you see on the oscilloscope's front panel, made up of 0, 1, X (don't care), R (rising edge), and F (falling edge) characters.
- When the format is HEX, the string begins with "0x" and contains hex digit characters or X (don't care for all four bits in the nibble).

With the hex format string, you can use the <edge\_source> and <edge> parameters to specify an edge on one of the channels.

**NOTE**

The optional <edge\_source> and <edge> parameters should be sent together or not at all. The edge can be specified in the ASCII <string> parameter. If the edge source and edge parameters are used, they take precedence.

---

You can only specify an edge on one channel. When an edge is specified, the :TRIGger:PATTERn:QUALifier does not apply.

**Query Syntax** :TRIGger:PATTERn?

The :TRIGger:PATTERn? query returns the pattern string, edge source, and edge.

**Return Format** <string>,<edge\_source>,<edge><NL>

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":TRIGger:PATTERn:FORMAT](#)" on page 1101
  - "[":TRIGger:PATTERn:QUALifier](#)" on page 1104
  - "[":TRIGger:MODE](#)" on page 1056

## :TRIGger:PATTERn:FORMAT

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:PATTERn:FORMAT <base>`  
                  `<base> ::= {ASCii | HEX}`

The :TRIGger:PATTERn:FORMAT command sets the entry (and query) number base used by the :TRIGger:PATTERn command. The default <base> is ASCii.

**Query Syntax**    `:TRIGger:PATTERn:FORMAT?`

The :TRIGger:PATTERn:FORMAT? query returns the currently set number base for pattern trigger patterns.

**Return Format**    `<base><NL>`  
                  `<base> ::= {ASC | HEX}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:PATTERn" on page 1099

## :TRIGger:PATTERn:GREaterthan

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:PATTERn:GREaterthan <greater_than_time>[<suffix>]`

`<greater_than_time>` ::= minimum trigger duration in seconds  
in NR3 format

`<suffix>` ::= {s | ms | us | ns | ps}

The :TRIGger:PATTERn:GREaterthan command sets the minimum duration for the defined pattern when :TRIGger:PATTERn:QUALifier is set to GREaterthan. The command also sets the timeout value when the :TRIGger:PATTERn:QUALifier is set to TImeout.

**Query Syntax**    `:TRIGger:PATTERn:GREaterthan?`

The :TRIGger:PATTERn:GREaterthan? query returns the minimum duration time for the defined pattern.

**Return Format**    `<greater_than_time><NL>`

**See Also**

- ["Introduction to :TRIGger Commands"](#) on page 1047
- [":TRIGger:PATTERn"](#) on page 1099
- [":TRIGger:PATTERn:QUALifier"](#) on page 1104
- [":TRIGger:MODE"](#) on page 1056

## :TRIGger:PATTERn:LESSthan

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:PATTERn:LESSthan <less\_than\_time>[<suffix>]  
<less\_than\_time> ::= maximum trigger duration in seconds  
in NR3 format  
<suffix> ::= {s | ms | us | ns | ps}

The :TRIGger:PATTERn:LESSthan command sets the maximum duration for the defined pattern when :TRIGger:PATTERn:QUALifier is set to LESSthan.

**Query Syntax** :TRIGger:PATTERn:LESSthan?

The :TRIGger:PATTERn:LESSthan? query returns the duration time for the defined pattern.

**Return Format** <less\_than\_time><NL>

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:PATTERn](#)" on page 1099
- "[:TRIGger:PATTERn:QUALifier](#)" on page 1104
- "[:TRIGger:MODE](#)" on page 1056

## :TRIGger:PATTERn:QUALifier

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:TRIGger:PATTERn:QUALifier &lt;qualifier&gt;</code>
	<code>&lt;qualifier&gt; ::= {ENTERed   GREaterthan   LESSthan   INRange   OUTRange   TIMEout}</code>

The :TRIGger:PATTERn:QUALifier command qualifies when the trigger occurs:

- ENTERed – when the pattern is entered.
- LESSthan – when the pattern is present for less than a time value.
- GREaterthan – when the pattern is present for greater than a time value. The trigger occurs when the pattern exits (not when the GREaterthan time value is exceeded).
- TIMEout – when the pattern is present for greater than a time value. In this case, the trigger occurs when the GREaterthan time value is exceeded (not when the pattern exits).
- INRange – when the pattern is present for a time within a range of values.
- OUTRange – when the pattern is present for a time outside of range of values.

Pattern durations are evaluated using a timer. The timer starts on the last edge that makes the pattern (logical AND) true. Except when the TIMEout qualifier is selected, the trigger occurs on the first edge that makes the pattern false, provided the time qualifier criteria has been met.

Set the GREaterthan qualifier value with the :TRIGger:PATTERn:GREaterthan command.

Set the LESSthan qualifier value with the :TRIGger:PATTERn:LESSthan command.

Set the INRange and OUTRange qualifier values with the :TRIGger:PATTERn:RANGE command.

Set the TIMEout qualifier value with the :TRIGger:PATTERn:GREaterthan command.

<b>Query Syntax</b>	<code>:TRIGger:PATTERn:QUALifier?</code>
---------------------	------------------------------------------

The :TRIGger:PATTERn:QUALifier? query returns the trigger duration qualifier.

<b>Return Format</b>	<code>&lt;qualifier&gt;&lt;NL&gt;</code>
----------------------	------------------------------------------

<b>See Also</b>	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>• "<a href="#">:TRIGger:PATTERn:GREaterthan</a>" on page 1102</li> <li>• "<a href="#">:TRIGger:PATTERn:LESSthan</a>" on page 1103</li> <li>• "<a href="#">:TRIGger:PATTERn:RANGE</a>" on page 1105</li> </ul>
-----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :TRIGger:PATTERn:RANGE

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:PATTERn:RANGE <less_than_time>[<suffix>], <greater_than_time>[<suffix>]`

`<greater_than_time> ::= 10 ns to 9.99 seconds in NR3 format`

`<less_than_time> ::= 15 ns to 10 seconds in NR3 format`

`<suffix> ::= {s | ms | us | ns | ps}`

The :TRIGger:PATTERn:RANGE command sets the duration for the defined pattern when the :TRIGger:PATTERn:QUALifier command is set to INRange or OUTRange. You can enter the parameters in any order – the smaller value becomes the <greater\_than\_time> and the larger value becomes the <less\_than\_time>.

**Query Syntax**    `:TRIGger:PATTERn:RANGE?`

The :TRIGger:PATTERn:RANGE? query returns the duration time for the defined pattern.

**Return Format**    `<less_than_time>, <greater_than_time><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:PATTERn"](#) on page 1099
- "[":TRIGger:PATTERn:QUALifier"](#) on page 1104
- "[":TRIGger:MODE"](#) on page 1056

## :TRIGger:RUNT Commands

**Table 147** :TRIGger:RUNT Commands Summary

Command	Query	Options and Query Returns
:TRIGger:RUNT:POLarit y <polarity> (see <a href="#">page 1107</a> )	:TRIGger:RUNT:POLarit y? (see <a href="#">page 1107</a> )	<polarity> ::= {POSitive   NEGative   EITHer}
:TRIGger:RUNT:QUALifi er <qualifier> (see <a href="#">page 1108</a> )	:TRIGger:RUNT:QUALifi er? (see <a href="#">page 1108</a> )	<qualifier> ::= {GREaterthan   LESSthan   NONE}
:TRIGger:RUNT:SOURce <source> (see <a href="#">page 1109</a> )	:TRIGger:RUNT:SOURce? (see <a href="#">page 1109</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:RUNT:TIME <time>[suffix] (see <a href="#">page 1110</a> )	:TRIGger:RUNT:TIME? (see <a href="#">page 1110</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

## :TRIGger:RUNT:POLarity

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:RUNT:POLarity <polarity>`

`<polarity> ::= {POSitive | NEGative | EITHer}`

The :TRIGger:RUNT:POLarity command sets the polarity for the runt trigger:

- POSitive – positive runt pulses.
- NEGative – negative runt pulses.
- EITHer – either positive or negative runt pulses.

**Query Syntax**    `:TRIGger:RUNT:POLarity?`

The :TRIGger:RUNT:POLarity? query returns the runt trigger polarity.

**Return Format**    `<polarity><NL>`

`<polarity> ::= {POS | NEG | EITH}`

**See Also**    · "[Introduction to :TRIGger Commands](#)" on page 1047

- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger:LEVel:HIGH](#)" on page 1054
- "[:TRIGger:LEVel:LOW](#)" on page 1055
- "[:TRIGger:RUNT:SOURce](#)" on page 1109

**:TRIGger:RUNT:QUALifier****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:RUNT:QUALifier <qualifier>``<qualifier> ::= {GREaterthan | LESSthan | NONE}`

The :TRIGger:RUNT:QUALifier command selects the qualifier used for specifying runt pulse widths:

- GREaterthan – triggers on runt pulses whose width is greater than the :TRIGger:RUNT:TIME.
- LESSthan – triggers on runt pulses whose width is less than the :TRIGger:RUNT:TIME.
- NONE – triggers on runt pulses of any width.

**Query Syntax**    `:TRIGger:RUNT:QUALifier?`

The :TRIGger:RUNT:QUALifier? query returns the runt trigger qualifier setting.

**Return Format**    `<qualifier><NL>``<qualifier> ::= {GRE | LESS NONE}`**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:MODE"](#) on page 1056
- "[":TRIGger:RUNT:TIME"](#) on page 1110

**:TRIGger:RUNT:SOURce**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:RUNT:SOURce <source>`

`<source> ::= CHANnel<n>`

`<n> ::= 1 to (# analog channels) in NR1 format`

The :TRIGger:RUNT:SOURce command selects the channel used to produce the trigger.

**Query Syntax**    `:TRIGger:RUNT:SOURce?`

The :TRIGger:RUNT:SOURce? query returns the current runt trigger source.

**Return Format**    `<source><NL>`

`<source> ::= CHAN<n>`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- "[:TRIGger:RUNT:POLarity](#)" on page 1107

**:TRIGger:RUNT:TIME****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:RUNT:TIME <time>[suffix]`    `<time> ::= floating-point number in NR3 format`    `[suffix] ::= {s | ms | us | ns | ps}`

When triggering on runt pulses whose width is greater than or less than a certain value (see :TRIGger:RUNT:QUALifier), the :TRIGger:RUNT:TIME command specifies the time used with the qualifier.

**Query Syntax**    `:TRIGger:RUNT:TIME?`

The :TRIGger:RUNT:TIME? query returns the current runt pulse qualifier time setting.

**Return Format**    `<time><NL>`    `<time> ::= floating-point number in NR3 format`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:RUNT:QUALifier" on page 1108

## :TRIGger:SHOLD Commands

**Table 148:** :TRIGger:SHOLD Commands Summary

Command	Query	Options and Query Returns
:TRIGger:SHOLD:SLOPe <slope> (see <a href="#">page 1112</a> )	:TRIGger:SHOLD:SLOPe? (see <a href="#">page 1112</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:SHOLD:SOURce :CLOCk <source> (see <a href="#">page 1113</a> )	:TRIGger:SHOLD:SOURce :CLOCk? (see <a href="#">page 1113</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:SHOLD:SOURce :DATA <source> (see <a href="#">page 1114</a> )	:TRIGger:SHOLD:SOURce :DATA? (see <a href="#">page 1114</a> )	<source> ::= {CHANnel<n>   DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:SHOLD:TIME:H OLD <time>[suffix] (see <a href="#">page 1115</a> )	:TRIGger:SHOLD:TIME:H OLD? (see <a href="#">page 1115</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}
:TRIGger:SHOLD:TIME:S ETup <time>[suffix] (see <a href="#">page 1116</a> )	:TRIGger:SHOLD:TIME:S ETup? (see <a href="#">page 1116</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**:TRIGger:SHOLD:SLOPe****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:SHOLD:SLOPe <slope>`  
                  `<slope> ::= {NEGative | POSitive}`

The :TRIGger:SHOLD:SLOPe command specifies whether the rising edge or the falling edge of the clock signal is used.

**Query Syntax**    `:TRIGger:SHOLD:SLOPe?`

The :TRIGger:SHOLD:SLOPe? query returns the current rising or falling edge setting.

**Return Format**    `<slope><NL>`  
                  `<slope> ::= {NEG | POS}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":TRIGger:SHOLD:SOURce:CLOCK" on page 1113
- ":TRIGger:SHOLD:SOURce:DATA" on page 1114

## :TRIGger:SHOLD:SOURce:CLOCK

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:TRIGger:SHOLD:SOURce:CLOCK &lt;source&gt;</code>
	<code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;}</code>
	<code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>
	<code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :TRIGger:SHOLD:SOURce:CLOCK command selects the input channel probing the clock signal.
<b>Query Syntax</b>	<code>:TRIGger:SHOLD:SOURce:CLOCK?</code>
	The :TRIGger:SHOLD:SOURce:CLOCK? query returns the currently set clock signal source.
<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>
	<code>&lt;source&gt; ::= {CHAN&lt;n&gt;   DIG&lt;d&gt;}</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">Introduction to :TRIGger Commands</a>" on page 1047</li> <li>· "<a href="#">:TRIGger:MODE</a>" on page 1056</li> <li>· "<a href="#">:TRIGger:SHOLD:SLOPe</a>" on page 1112</li> </ul>

**:TRIGger:SHOLD:SOURce:DATA****N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:TRIGger:SHOLD:SOURce:DATA &lt;source&gt;</code>  <code>&lt;source&gt; ::= {CHANnel&lt;n&gt;   DIGital&lt;d&gt;}</code>  <code>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</code>  <code>&lt;d&gt; ::= 0 to (# digital channels - 1) in NR1 format</code>
	The :TRIGger:SHOLD:SOURce:DATA command selects the input channel probing the data signal.
<b>Query Syntax</b>	<code>:TRIGger:SHOLD:SOURce:DATA?</code>
	The :TRIGger:SHOLD:SOURce:DATA? query returns the currently set data signal source.
<b>Return Format</b>	<code>&lt;source&gt;&lt;NL&gt;</code>  <code>&lt;source&gt; ::= {CHAN&lt;n&gt;   DIG&lt;d&gt;}</code>
<b>See Also</b>	<ul style="list-style-type: none"><li><a href="#">"Introduction to :TRIGger Commands"</a> on page 1047</li><li><a href="#">":TRIGger:MODE"</a> on page 1056</li><li><a href="#">":TRIGger:SHOLD:SLOPe"</a> on page 1112</li></ul>

**:TRIGger:SHOLD:TIME:HOLD**

**N** (see [page 1334](#))

**Command Syntax**    :TRIGger:SHOLD:TIME:HOLD <time>[suffix]  
                          <time> ::= floating-point number in NR3 format  
                          [suffix] ::= {s | ms | us | ns | ps}

The :TRIGger:SHOLD:TIME:HOLD command sets the hold time.

**Query Syntax**    :TRIGger:SHOLD:TIME:HOLD?

The :TRIGger:SHOLD:TIME:HOLD? query returns the currently specified hold time.

**Return Format**    <time><NL>  
                          <time> ::= floating-point number in NR3 format

**See Also**    • "Introduction to :TRIGger Commands" on page 1047

**:TRIGger:SHOLD:TIME:SETup****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:SHOLD:TIME:SETup <time>[suffix]`  
`<time> ::= floating-point number in NR3 format`  
`[suffix] ::= {s | ms | us | ns | ps}`

The :TRIGger:SHOLD:TIME:SETup command sets the setup time.

**Query Syntax**    `:TRIGger:SHOLD:TIME:SETup?`  
The :TRIGger:SHOLD:TIME:SETup? query returns the currently specified setup time.

**Return Format**    `<time><NL>`  
`<time> ::= floating-point number in NR3 format`

**See Also**    · "Introduction to :TRIGger Commands" on page 1047

## :TRIGger:TRANSition Commands

The :TRIGger:TRANSition commands set the rise/fall time trigger options.

**Table 149**:TRIGger:TRANSition Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TRANSition:QUALifier <qualifier> (see <a href="#">page 1118</a> )	:TRIGger:TRANSition:QUALifier? (see <a href="#">page 1118</a> )	<qualifier> ::= {GREaterthan   LESSthan}
:TRIGger:TRANSition:SLOPe <slope> (see <a href="#">page 1119</a> )	:TRIGger:TRANSition:SLOPe? (see <a href="#">page 1119</a> )	<slope> ::= {NEGative   POSitive}
:TRIGger:TRANSition:SOURce <source> (see <a href="#">page 1120</a> )	:TRIGger:TRANSition:SOURce? (see <a href="#">page 1120</a> )	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TRANSition:TIME <time>[suffix] (see <a href="#">page 1121</a> )	:TRIGger:TRANSition:TIME? (see <a href="#">page 1121</a> )	<time> ::= floating-point number in NR3 format [suffix] ::= {s   ms   us   ns   ps}

**:TRIGger:TRANSition:QUALifier****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TRANSition:QUALifier <qualifier>`  
`<qualifier> ::= {GREaterthan | LESSthan}`

The :TRIGger:TRANSition:QUALifier command specifies whether you are looking for rise/fall times greater than or less than a certain time value. The time value is set using the :TRIGger:TRANSition:TIME command.

**Query Syntax**    `:TRIGger:TRANSition:QUALifier?`

The :TRIGger:TRANSition:QUALifier? query returns the current rise/fall time trigger qualifier setting.

**Return Format**    `<qualifier><NL>`  
`<qualifier> ::= {GRE | LESS}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:TRANSition:TIME" on page 1121
- ":TRIGger:MODE" on page 1056

**:TRIGger:TRANSition:SLOPe**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TRANSition:SLOPe <slope>`  
`<slope> ::= {NEGative | POSitive}`

The :TRIGger:TRANSition:SLOPe command specifies a POSitive rising edge or a NEGative falling edge.

**Query Syntax**    `:TRIGger:TRANSition:SLOPe?`

The :TRIGger:TRANSition:SLOPe? query returns the current rise/fall time trigger slope setting.

**Return Format**    `<slope><NL>`  
`<slope> ::= {NEG | POS}`

**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":TRIGger:TRANSition:SOURce" on page 1120

**:TRIGger:TRANSition:SOURce****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:TRANSition:SOURce <source>`    `<source> ::= CHANnel<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :TRIGger:TRANSition:SOURce command selects the channel used to produce the trigger.

**Query Syntax**    `:TRIGger:TRANSition:SOURce?`

The :TRIGger:TRANSition:SOURce? query returns the current transition trigger source.

**Return Format**    `<source><NL>`    `<source> ::= CHAN<n>`**See Also**

- "Introduction to :TRIGger Commands" on page 1047
- ":TRIGger:MODE" on page 1056
- ":TRIGger:TRANSition:SLOPe" on page 1119

## :TRIGger:TRANSition:TIME

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TRANSition:TIME <time>[suffix]`  
`<time> ::= floating-point number in NR3 format`  
`[suffix] ::= {s | ms | us | ns | ps}`

The :TRIGger:TRANSition:TIME command sets the time value for rise/fall time triggers. You also use the :TRIGger:TRANSition:QUALifier command to specify whether you are triggering on times greater than or less than this time value.

**Query Syntax**    `:TRIGger:TRANSition:TIME?`

The :TRIGger:TRANSition:TIME? query returns the current rise/fall time trigger time value.

**Return Format**    `<time><NL>`  
`<time> ::= floating-point number in NR3 format`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:TRANSition:QUALifier](#)" on page 1118

## :TRIGger:TV Commands

**Table 150:**TRIGger:TV Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TV:LINE <line number> (see <a href="#">page 1123</a> )	:TRIGger:TV:LINE? (see <a href="#">page 1123</a> )	<line number> ::= integer in NR1 format
:TRIGger:TV:MODE <tv mode> (see <a href="#">page 1124</a> )	:TRIGger:TV:MODE? (see <a href="#">page 1124</a> )	<tv mode> ::= {FIELD1   FIELD2   AFields   ALINes   LINE   LFIELD1   LFIELD2   LATernate}
:TRIGger:TV:POLarity <polarity> (see <a href="#">page 1125</a> )	:TRIGger:TV:POLarity? (see <a href="#">page 1125</a> )	<polarity> ::= {POSitive   NEGative}
:TRIGger:TV:SOURce <source> (see <a href="#">page 1126</a> )	:TRIGger:TV:SOURce? (see <a href="#">page 1126</a> )	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TV:STANDARD <standard> (see <a href="#">page 1127</a> )	:TRIGger:TV:STANDARD? (see <a href="#">page 1127</a> )	<standard> ::= {NTSC   PAL   PALM   SECam} <standard> ::= {GENeric   {P480L60HZ   P480}   {P720L60HZ   P720}   {P1080L24HZ   P1080}   P1080L25HZ   P1080L50HZ   P1080L60HZ   {I1080L50HZ   I1080}   I1080L60HZ} with extended video triggering license
:TRIGger:TV:UDTV:ENUM ber <count> (see <a href="#">page 1128</a> )	:TRIGger:TV:UDTV:ENUM ber? (see <a href="#">page 1128</a> )	<count> ::= edge number in NR1 format
:TRIGger:TV:UDTV:HSYN C {{0   OFF}   {1   ON}} (see <a href="#">page 1129</a> )	:TRIGger:TV:UDTV:HSYN C? (see <a href="#">page 1129</a> )	{0   1}
:TRIGger:TV:UDTV:HTIM E <time> (see <a href="#">page 1130</a> )	:TRIGger:TV:UDTV:HTIM E? (see <a href="#">page 1130</a> )	<time> ::= seconds in NR3 format
:TRIGger:TV:UDTV:PGTH AN <min_time> (see <a href="#">page 1131</a> )	:TRIGger:TV:UDTV:PGTH AN? (see <a href="#">page 1131</a> )	<min_time> ::= seconds in NR3 format

## :TRIGger:TV:LINE

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:TV:LINE <line\_number>

<line\_number> ::= integer in NR1 format

The :TRIGger:TV:LINE command allows triggering on a specific line of video. The line number limits vary with the standard and mode, as shown in the following table.

**Table 151** TV Trigger Line Number Limits

TV Standard	Mode				
	LINE	LField1	LField2	LALTernate	VERTical
NTSC		1 to 263	1 to 262	1 to 262	
PAL		1 to 313	314 to 625	1 to 312	
PAL-M		1 to 263	264 to 525	1 to 262	
SECAM		1 to 313	314 to 625	1 to 312	
GENERIC		1 to 1024	1 to 1024		1 to 1024
P480L60HZ	1 to 525				
P720L60HZ	1 to 750				
P1080L24HZ	1 to 1125				
P1080L25HZ	1 to 1125				
P1080L50HZ	1 to 1125				
P1080L60HZ	1 to 1125				
I1080L50HZ	1 to 1125				
I1080L60HZ	1 to 1125				

**Query Syntax** :TRIGger:TV:LINE?

The :TRIGger:TV:LINE? query returns the current TV trigger line number setting.

**Return Format** <line\_number><NL>

<line\_number> ::= integer in NR1 format

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[":TRIGger:TV:STANDARD](#)" on page 1127
  - "[":TRIGger:TV:MODE](#)" on page 1124

## :TRIGger:TV:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TV:MODE <mode>`

```
<mode> ::= {FIEld1 | FIEld2 | AFIElds | ALINes | LINE | LFIeld1  
| LFIeld2 | LALTernate}
```

The :TRIGger:TV:MODE command selects the TV trigger mode and field. The LALTernate parameter is not available when :TRIGger:TV:STANDARD is GENeric.

Old forms for <mode> are accepted:

<mode>	Old Forms Accepted
FIEld1	F1
FIEld2	F2
AFIElds	ALLFields, ALLFLDS
ALINes	ALLLines
LFIeld1	LINEF1, LINEFIELD1
LFIeld2	LINEF2, LINEFIELD2
LALTernate	LINEAlt

**Query Syntax**    `:TRIGger:TV:MODE?`

The :TRIGger:TV:MODE? query returns the TV trigger mode.

**Return Format**    `<value><NL>`

```
<value> ::= {FIE1 | FIE2 | AFI | ALIN | LINE | LFI1 | LFI2 | LALT}
```

- See Also**
- "[Introduction to :TRIGger Commands](#)" on page 1047
  - "[:TRIGger:TV:STANDARD](#)" on page 1127
  - "[:TRIGger:MODE](#)" on page 1056

## :TRIGger:TV:POLarity

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:TV:POLarity <polarity>

<polarity> ::= {POSITIVE | NEGATIVE}

The :TRIGger:TV:POLarity command sets the polarity for the TV trigger.

**Query Syntax** :TRIGger:TV:POLarity?

The :TRIGger:TV:POLarity? query returns the TV trigger polarity.

**Return Format** <polarity><NL>

<polarity> ::= {POS | NEG}

**See Also** • ["Introduction to :TRIGger Commands"](#) on page 1047

• [":TRIGger:MODE"](#) on page 1056

• [":TRIGger:TV:SOURce"](#) on page 1126

**:TRIGger:TV:SOURce****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:TV:SOURce <source>`

```
<source> ::= {CHANnel<n>}  
<n> ::= 1 to (# analog channels) in NR1 format
```

The :TRIGger:TV:SOURce command selects the channel used to produce the trigger.

**Query Syntax**    `:TRIGger:TV:SOURce?`

The :TRIGger:TV:SOURce? query returns the current TV trigger source.

**Return Format**    `<source><NL>`

```
<source> ::= {CHAN<n>}
```

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger:TV:POLarity](#)" on page 1125

**Example Code**

- "[Example Code](#)" on page 1076

## :TRIGger:TV:STANDARD

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:TV:STANDARD <standard>

```
<standard> ::= {GENeric | NTSC | PALM | PAL | SECam
                 | {P480L60HZ | P480} | {P720L60HZ | P720}
                 | {P1080L24HZ | P1080} | P1080L25HZ
                 | P1080L50HZ | P1080L60HZ
                 | {I1080L50HZ | I1080} | I1080L60HZ}
```

The :TRIGger:TV:STANDARD command selects the video standard:

- NTSC
- PAL
- PAL-M
- SECAM

With an extended Video triggering license, the oscilloscope additionally supports these standards:

- Generic – GENeric mode is non-interlaced.
- EDTV 480p/60
- HDTV 720p/60
- HDTV 1080p/24
- HDTV 1080p/25
- HDTV 1080i/50
- HDTV 1080i/60

**Query Syntax** :TRIGger:TV:STANDARD?

The :TRIGger:TV:STANDARD? query returns the current TV trigger standard setting.

**Return Format** <standard><NL>

```
<standard> ::= {GEN | NTSC | PALM | PAL | SEC | P480L60HZ | P760L60HZ
                 | P1080L24HZ | P1080L25HZ | P1080L50HZ | P1080L60HZ
                 | I1080L50HZ | I1080L60HZ}
```

**:TRIGger:TV:UDTV:ENUMber****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TV:UDTV:ENUMber <count>`  
                  `<count> ::= edge number in NR1 format`

The :TRIGger:TV:UDTV:ENUMber command specifies the Generic video trigger's Nth edge to trigger on after synchronizing with the vertical sync.

**Query Syntax**    `:TRIGger:TV:UDTV:ENUMber?`

The :TRIGger:TV:UDTV:ENUMber query returns the edge count setting.

**Return Format**    `<count><NL>`  
                  `<count> ::= edge number in NR1 format`

**See Also**

- "[:TRIGger:TV:STANDARD](#)" on page 1127
- "[:TRIGger:TV:UDTV:PGTHan](#)" on page 1131
- "[:TRIGger:TV:UDTV:HSYNC](#)" on page 1129

## :TRIGger:TV:UDTV:HSync

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:TV:UDTV:HSync {{0 | OFF} | {1 | ON}}

The :TRIGger:TV:UDTV:HSync command enables or disables the horizontal sync control in the Generic video trigger.

For interleaved video, enabling the HSync control and setting the HTIME adjustment to the sync time of the probed video signal allows the ENUMber function to count only lines and not double count during equalization. Additionally, the Field Holdoff can be adjusted so that the oscilloscope triggers once per frame.

Similarly, for progressive video with a tri-level sync, enabling the HSync control and setting the HTIME adjustment to the sync time of the probed video signal allows the ENUMber function to count only lines and not double count during vertical sync.

**Query Syntax** :TRIGger:TV:UDTV:HSync?

The :TRIGger:TV:UDTV:HSync query returns the horizontal sync control setting.

**Return Format** {0 | 1}

- See Also**
- "[:TRIGger:TV:STANDARD](#)" on page 1127
  - "[:TRIGger:TV:UDTV:HTIME](#)" on page 1130
  - "[:TRIGger:TV:UDTV:ENUMber](#)" on page 1128
  - "[:TRIGger:TV:UDTV:PGTHan](#)" on page 1131

**:TRIGger:TV:UDTV:HTIMe****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TV:UDTV:HTIMe <time>`  
                      `<time> ::= seconds in NR3 format`

When the Generic video trigger's horizontal sync control is enabled, the :TRIGger:TV:UDTV:HTIMe command sets the minimum time the horizontal sync pulse must be present to be considered valid.

**Query Syntax**    `:TRIGger:TV:UDTV:HTIMe?`  
The :TRIGger:TV:UDTV:HTIMe query returns the horizontal sync time setting.

**Return Format**    `<time><NL>`  
                      `<time> ::= seconds in NR3 format`

**See Also**

- [":TRIGger:TV:STANDARD"](#) on page 1127
- [":TRIGger:TV:UDTV:HSYNC"](#) on page 1129

**:TRIGger:TV:UDTV:PGTHan**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:TV:UDTV:PGTHan <min_time>`  
`<min_time> ::= seconds in NR3 format`

The :TRIGger:TV:UDTV:PGTHan command specifies the "greater than the sync pulse width" time in the Generic video trigger. This setting allows oscilloscope synchronization to the vertical sync.

**Query Syntax**    `:TRIGger:TV:UDTV:PGTHan?`

The :TRIGger:TV:UDTV:PGTHan query returns the "greater than the sync pulse width" time setting.

**Return Format**    `<min_time><NL>`  
`<min_time> ::= seconds in NR3 format`

**See Also**

- "[:TRIGger:TV:STANDARD](#)" on page 1127
- "[:TRIGger:TV:UDTV:ENUMber](#)" on page 1128
- "[:TRIGger:TV:UDTV:HSYNC](#)" on page 1129

## :TRIGger:USB Commands

**Table 152:**TRIGger:USB Commands Summary

Command	Query	Options and Query Returns
:TRIGger:USB:SOURce:D MINus <source> (see <a href="#">page 1133</a> )	:TRIGger:USB:SOURce:D MINus? (see <a href="#">page 1133</a> )	<source> ::= {CHANnel<n>   EXTernal} for the DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:USB:SOURce:D PLus <source> (see <a href="#">page 1134</a> )	:TRIGger:USB:SOURce:D PLus? (see <a href="#">page 1134</a> )	<source> ::= {CHANnel<n>   EXTernal} for the DSO models <source> ::= {CHANnel<n>   DIGItal<d>} for the MSO models <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:USB:SPEed <value> (see <a href="#">page 1135</a> )	:TRIGger:USB:SPEed? (see <a href="#">page 1135</a> )	<value> ::= {LOW   FULL}
:TRIGger:USB:TRIGger <value> (see <a href="#">page 1136</a> )	:TRIGger:USB:TRIGger? (see <a href="#">page 1136</a> )	<value> ::= {SOP   EOP   ENTerSuspend   EXITsuspend   RESet}

## :TRIGger:USB:SOURce:DMINus

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:USB:SOURce:DMINus <source>`

```

<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format

```

The :TRIGger:USB:SOURce:DMINus command sets the source for the USB D- signal.

**Query Syntax**    `:TRIGger:USB:SOURce:DMINus?`

The :TRIGger:USB:SOURce:DMINus? query returns the current source for the USB D- signal.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger:USB:SOURce:DPLus](#)" on page 1134
- "[:TRIGger:USB:TRIGger](#)" on page 1136

**:TRIGger:USB:SOURce:DPLus**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:USB:SOURce:DPLus <source>`

```
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :TRIGger:USB:SOURce:DPLus command sets the source for the USB D+ signal.

**Query Syntax**    `:TRIGger:USB:SOURce:DPLus?`

The :TRIGger:USB:SOURce:DPLus? query returns the current source for the USB D+ signal.

**Return Format**    `<source><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger:USB:SOURce:DMINus](#)" on page 1133
- "[:TRIGger:USB:TRIGger](#)" on page 1136

**:TRIGger:USB:SPEed**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:USB:SPEed <value>`  
                  `<value> ::= {LOW | FULL}`

The :TRIGger:USB:SPEed command sets the expected USB signal speed to be Low Speed (1.5 Mb/s) or Full Speed (12 Mb/s).

**Query Syntax**    `:TRIGger:USB:SPEed?`

The :TRIGger:USB:SPEed? query returns the current speed value for the USB signal.

**Return Format**    `<value><NL>`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:TRIGger:MODE](#)" on page 1056
- "[:TRIGger:USB:SOURce:DMINus](#)" on page 1133
- "[:TRIGger:USB:SOURce:DPLus](#)" on page 1134
- "[:TRIGger:USB:TRIGger](#)" on page 1136

**:TRIGger:USB:TRIGger**

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:USB:TRIGger <value>`

`<value> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet}`

The :TRIGger:USB:TRIGger command sets where the USB trigger will occur:

- SOP – Start of packet.
- EOP – End of packet.
- ENTersuspend – Enter suspend state.
- EXITsuspend – Exit suspend state.
- RESet – Reset complete.

**Query Syntax**    `:TRIGger:USB:TRIGger?`

The :TRIGger:USB:TRIGger? query returns the current USB trigger value.

**Return Format**    `<value><NL>`

`<value> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet}`

**See Also**    • "[Introduction to :TRIGger Commands](#)" on page 1047

    • "[:TRIGger:MODE](#)" on page 1056

    • "[:TRIGger:USB:SPEed](#)" on page 1135

## :TRIGger:ZONE Commands

**Table 153:** :TRIGger:ZONE Commands Summary

Command	Query	Options and Query Returns
:TRIGger:ZONE:SOURce <source> (see page 1138)	:TRIGger:ZONE:SOURce? (see page 1138)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:ZONE:STATE { {0   OFF}   {1   ON} } (see page 1139)	:TRIGger:ZONE:STATE? (see page 1139)	{0   1}
:TRIGger:ZONE<n>:MODE <mode> (see page 1140)	:TRIGger:ZONE<n>:MODE? (see page 1140)	<mode> ::= {INTersect   NOTintersect} <n> ::= 1-2 in NR1 format
:TRIGger:ZONE<n>:PLACEMENT <width>, <height>, <x_center>, <y_center> (see page 1141)	:TRIGger:ZONE<n>:PLACEMENT? (see page 1141)	<width> ::= width of zone in seconds <height> ::= height of zone in volts <x_center> ::= center of zone in seconds <y_center> ::= center of zone in volts <n> ::= 1-2 in NR1 format
n/a	:TRIGger:ZONE<n>:VALIDITY? (see page 1142)	<value> ::= {VALid   INValid   OSCreen} <n> ::= 1-2 in NR1 format
:TRIGger:ZONE<n>:STATE { {0   OFF}   {1   ON} } (see page 1143)	:TRIGger:ZONE<n>:STATE? (see page 1143)	{0   1} <n> ::= 1-2 in NR1 format

**:TRIGger:ZONE:SOURce****N** (see [page 1334](#))**Command Syntax**    `:TRIGger:ZONE:SOURce <source>`    `<source> ::= {CHANnel<n>}`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :TRIGger:ZONE:SOURce command sets the analog source channel shared by all zones.

**Query Syntax**    `:TRIGger:ZONE:SOURce?`

The :TRIGger:ZONE:SOURce? query returns the analog source channel specified for zone qualified triggers.

**Return Format**    `<source><NL>`    `<source> ::= {CHAN<n>}`**See Also**    • [":TRIGger:ZONE:STATe"](#) on page 1139

## :TRIGger:ZONE:STATe

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:ZONE:STATe <on\_off>

<on\_off> ::= {{0 | OFF} | {1 | ON}}

The :TRIGger:ZONE:STATe command enables or disables the zone qualified trigger feature.

When the zone qualified trigger is on, the zone(s) are actively being used to qualify the trigger.

Note that the :TRIGger:ZONE<n>:STATe setting must also be ON for a zone to be active.

Note that :TRIGger:ZONE:STATe mimics the behavior of the **[Zone]** key on the front panel, and :TRIGger:ZONE<n>:STATe mimics the behavior of the **Zone 1 On** and **Zone 2 On** softkeys. At least one zone's state must be on for the Zone Trigger feature (:TRIGger:ZONE:STATe) to be on. When the states of both individual zones are turned off, Zone Trigger is automatically turned off. In this case, when Zone Trigger is turned back on Zone 1 is forced to on. Otherwise, if at least one zone was on when Zone Trigger was turned off, the same configuration of individual zone on/off states will be restored when Zone Trigger is turned back on.

**Query Syntax** :TRIGger:ZONE:STATe?

The :TRIGger:ZONE:STATe? query returns whether the zone qualified trigger feature is enabled or disabled.

**Return Format** <on\_off><NL>

<on\_off> ::= {0 | 1}

**See Also**

- "[:TRIGger:ZONE<n>:STATE](#)" on page 1143
- "[:TRIGger:ZONE:SOURce](#)" on page 1138

**:TRIGger:ZONE<n>:MODE****N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:ZONE<n>:MODE <mode>`  
                  `<mode> ::= {INTersect | NOTintersect}`  
                  `<n> ::= 1-2 in NR1 format`

The :TRIGger:ZONE<n>:MODE command sets the zone qualifying condition for Zone 1 or Zone 2 as either "Must Intersect" or "Must Not Intersect".

**Query Syntax**    `:TRIGger:ZONE<n>:MODE?`

The :TRIGger:ZONE<n>:MODE? query returns the zone qualifying condition for Zone 1 or Zone 2.

**Return Format**    `<mode><NL>`  
                  `<mode> ::= {INT | NOT}`

**See Also**

- [":TRIGger:ZONE<n>:STATE" on page 1143](#)
- [":TRIGger:ZONE<n>:PLACEMENT" on page 1141](#)
- [":TRIGger:ZONE<n>:VALIDITY" on page 1142](#)

## :TRIGger:ZONE<n>:PLACement

**N** (see [page 1334](#))

**Command Syntax** :TRIGger:ZONE<n>:PLACement <width>, <height>, <x\_center>, <y\_center>

<width> ::= width of zone in seconds

<height> ::= height of zone in volts

<x\_center> ::= center of zone in seconds

<y\_center> ::= center of zone in volts

<n> ::= 1-2 in NR1 format

The :TRIGger:ZONE<n>:PLACement command sets the size and location of Zone 1 or Zone 2.

No error is returned if the zone is placed off-screen, or if the zones overlap such that Zone 2 becomes invalid. The :TRIGger:ZONE<n>:VALidity? query is used to retrieve this information.

**Query Syntax** :TRIGger:ZONE<n>:PLACement?

The :TRIGger:ZONE<n>:PLACement? query returns the size and location of Zone 1 or Zone 2.

**Return Format** <opt><NL>

<opt> ::= <width>, <height>, <x\_center>, <y\_center>

**See Also**

- "[:TRIGger:ZONE<n>:STATE](#)" on page 1143
- "[:TRIGger:ZONE<n>:MODE](#)" on page 1140
- "[:TRIGger:ZONE<n>:VALidity](#)" on page 1142

**:TRIGger:ZONE<n>:VALidity**

**N** (see [page 1334](#))

**Query Syntax**    `:TRIGger:ZONE<n>:VALidity?`

`<n>` ::= 1-2 in NRI format

The `:TRIGger:ZONE<n>:VALidity?` query returns the validity of Zone 1 or Zone 2.

- INValid is returned (for Zone 2 only) when Zone 1 and Zone 2 overlap and have opposing qualifying conditions (modes). Zone 1 can never be invalid.
- OSCReen (off-screen) is returned when the associated zone is off-screen, and thus not being used to qualify the trigger.
- A zone is valid when it is neither invalid nor off-screen.

The validity of a zone is not affected by the zone's state. For example, a zone can be valid and off. You cannot directly set the validity of a zone.

**Return Format**    `<validity><NL>`

`<validity>` ::= {VALid | INValid | OSCReen}

**See Also**    [":TRIGger:ZONE<n>:STATE"](#) on page 1143  
[":TRIGger:ZONE<n>:MODE"](#) on page 1140  
[":TRIGger:ZONE<n>:PLACement"](#) on page 1141

## :TRIGger:ZONE<n>:STATE

**N** (see [page 1334](#))

**Command Syntax**    `:TRIGger:ZONE<n>:STATE <on_off>`

`<n> ::= {1 | 2}`

`<on_off> ::= {{0 | OFF} | {1 | ON}}`

`<n> ::= 1-2 in NR1 format`

The :TRIGger:ZONE<n>:STATE command sets the state for Zone 1 or Zone 2.

- When a zone's state is on, and the Zone Trigger feature is on (see "[:TRIGger:ZONE:STATE](#)" on page 1139), that zone is actively being used to qualify the trigger if it is not invalid or off-screen (see "[:TRIGger:ZONE<n>:VALidity](#)" on page 1142).
- When the Zone Trigger feature is off, no zones are being used to qualify the trigger, regardless of their individual states.

Note that :TRIGger:ZONE:STATE mimics the behavior of the **[Zone]** key on the front panel, and :TRIGger:ZONE<n>:STATE mimics the behavior of the **Zone 1 On** and **Zone 2 On** softkeys. At least one zone's state must be on for the Zone Trigger feature (:TRIGger:ZONE:STATE) to be on. When the states of both individual zones are turned off, Zone Trigger is automatically turned off. In this case, when Zone Trigger is turned back on Zone 1 is forced to on. Otherwise, if at least one zone was on when Zone Trigger was turned off, the same configuration of individual zone on/off states will be restored when Zone Trigger is turned back on.

**Query Syntax**    `:TRIGger:ZONE<n>:STATE?`

The :TRIGger:ZONE<n>:STATE? query returns the state of Zone 1 or Zone 2.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {0 | 1}`

**See Also**

- "[:TRIGger:ZONE:STATE](#)" on page 1139
- "[:TRIGger:ZONE<n>:VALidity](#)" on page 1142



## 34 :WAVeform Commands

Provide access to waveform data. See "[Introduction to :WAVeform Commands](#)" on page 1147.

**Table 154** :WAVeform Commands Summary

Command	Query	Options and Query Returns
:WAVeform:BYTeorder <value> (see <a href="#">page 1153</a> )	:WAVeform:BYTeorder? (see <a href="#">page 1153</a> )	<value> ::= {LSBFFirst   MSBFFirst}
n/a	:WAVeform:COUNT? (see <a href="#">page 1154</a> )	<count> ::= an integer from 1 to 65536 in NR1 format
n/a	:WAVeform:DATA? (see <a href="#">page 1155</a> )	<binary block length bytes>, <binary data> For example, to transmit 1000 bytes of data, the syntax would be: #800001000<1000 bytes of data><NL> 8 is the number of digits that follow 00001000 is the number of bytes to be transmitted <1000 bytes of data> is the actual data
:WAVeform:FORMAT <value> (see <a href="#">page 1157</a> )	:WAVeform:FORMAT? (see <a href="#">page 1157</a> )	<value> ::= {WORD   BYTE   ASCII}
:WAVeform:POINTS <# points> (see <a href="#">page 1158</a> )	:WAVeform:POINTS? (see <a href="#">page 1158</a> )	<# points> ::= {100   250   500   1000   <points_mode>} if waveform points mode is NORMAl <# points> ::= {100   250   500   1000   2000 ... 8000000 in 1-2-5 sequence   <points_mode>} if waveform points mode is MAXimum or RAW <points_mode> ::= {NORMAl   MAXimum   RAW}

**Table 154:** WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVEform:POINTs:MODE <points_mode> (see <a href="#">page 1160</a> )	:WAVEform:POINTs:MODE ? (see <a href="#">page 1160</a> )	<points_mode> ::= {NORMal   MAXimum   RAW}
n/a	:WAVEform:PREamble? (see <a href="#">page 1162</a> )	<p>&lt;preamble_block&gt; ::= &lt;format NR1&gt;, &lt;type NR1&gt;, &lt;points NR1&gt;, &lt;count NR1&gt;, &lt;xincrement NR3&gt;, &lt;xorigin NR3&gt;, &lt;xreference NR1&gt;, &lt;yincrement NR3&gt;, &lt;yorigin NR3&gt;, &lt;yreference NR1&gt;</p> <p>&lt;format&gt; ::= an integer in NR1 format:</p> <ul style="list-style-type: none"> <li>• 0 for BYTE format</li> <li>• 1 for WORD format</li> <li>• 2 for ASCII format</li> </ul> <p>&lt;type&gt; ::= an integer in NR1 format:</p> <ul style="list-style-type: none"> <li>• 0 for NORMAL type</li> <li>• 1 for PEAK detect type</li> <li>• 3 for AVERAGE type</li> <li>• 4 for HRESolution type</li> </ul> <p>&lt;count&gt; ::= Average count, or 1 if PEAK detect type or NORMAL; an integer in NR1 format</p>
n/a	:WAVEform:SEGmented:COUNT? (see <a href="#">page 1165</a> )	<count> ::= an integer from 2 to 1000 in NR1 format (with Option SGM)
n/a	:WAVEform:SEGmented:TAG? (see <a href="#">page 1166</a> )	<time_tag> ::= in NR3 format (with Option SGM)
:WAVEform:SOURce <source> (see <a href="#">page 1167</a> )	:WAVEform:SOURce? (see <a href="#">page 1167</a> )	<p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS} for DSO models</p> <p>&lt;source&gt; ::= {CHANnel&lt;n&gt;   POD{1   2}   BUS{1   2}   FUNCtion&lt;m&gt;   MATH&lt;m&gt;   FFT   SBUS} for MSO models</p> <p>&lt;n&gt; ::= 1 to (# analog channels) in NR1 format</p> <p>&lt;m&gt; ::= 1 to (# math functions) in NR1 format</p>
:WAVEform:SOURce:SUBS ource <subsource> (see <a href="#">page 1171</a> )	:WAVEform:SOURce:SUBS ource? (see <a href="#">page 1171</a> )	<subsource> ::= {{SUB0   RX   MOSI}   {SUB1   TX   MISO}}

**Table 154:** WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVEform:TYPE? (see <a href="#">page 1172</a> )	<return_mode> ::= {NORM   PEAK   AVER   HRES}
:WAVEform:UNSIGNED {{0   OFF}   {1   ON}} (see <a href="#">page 1173</a> )	:WAVEform:UNSIGNED? (see <a href="#">page 1173</a> )	{0   1}
:WAVEform:VIEW <view> (see <a href="#">page 1174</a> )	:WAVEform:VIEW? (see <a href="#">page 1174</a> )	<view> ::= {MAIN}
n/a	:WAVEform:XINCREMENT? (see <a href="#">page 1175</a> )	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVEform:XORIGIN? (see <a href="#">page 1176</a> )	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVEform:XREFERENCE? (see <a href="#">page 1177</a> )	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)
n/a	:WAVEform:YINCREMENT? (see <a href="#">page 1178</a> )	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVEform:YORIGIN? (see <a href="#">page 1179</a> )	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVEform:YREFERENCE? (see <a href="#">page 1180</a> )	<return_value> ::= y-reference value in the current preamble in NR1 format

**Introduction to :WAVEform Commands** The WAVEform subsystem is used to transfer data to a controller from the oscilloscope waveform memories. The queries in this subsystem will only operate when the channel selected by :WAVEform:SOURce is on.

### Waveform Data and Preamble

The waveform record is actually contained in two portions: the preamble and waveform data. The waveform record must be read from the oscilloscope by the controller using two separate commands, :WAVEform:DATA (see [page 1155](#)) and :WAVEform:PREamble (see [page 1162](#)). The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data, which includes the number of points acquired, the format of acquired data, and the type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data, so that word and byte data can be translated to time and voltage values.

## Data Acquisition Types

There are four types of waveform acquisitions that can be selected for analog channels with the :ACQuire:TYPE command (see [page 255](#)): NORMal, AVERage, PEAK, and HRESolution. Digital channels are always acquired using NORMal. When the data is acquired using the :DIGItize command (see [page 215](#)) or :RUN command (see [page 236](#)), the data is placed in the channel buffer of the specified source.

Once you have acquired data with the :DIGItize command, the instrument is stopped. If the instrument is restarted (via the programming interface or the front panel), or if any instrument setting is changed, the data acquired with the :DIGItize command may be overwritten. You should first acquire the data with the :DIGItize command, then immediately read the data with the :WAVEform:DATA? query (see [page 1155](#)) before changing any instrument setup.

A waveform record consists of either all of the acquired points or a subset of the acquired points. The number of points acquired may be queried using :ACQuire:POINts? (see [page 248](#)).

### **Helpful Hints:**

The number of points transferred to the computer is controlled using the :WAVEform:POINts command (see [page 1158](#)). If :WAVEform:POINts MAXimum is specified and the instrument is not running (stopped), all of the points that are displayed are transferred. This can be as many as 4,000,000 in some operating modes or as many as 8,000,000 for a digital channel on the mixed signal oscilloscope. Fewer points may be specified to speed data transfers and minimize controller analysis time. The :WAVEform:POINts may be varied even after data on a channel is acquired. However, this decimation may result in lost pulses and transitions. The number of points selected for transfer using :WAVEform:POINts must be an even divisor of 1,000 or be set to MAXimum. :WAVEform:POINts determines the increment between time buckets that will be transferred. If POINts = MAXimum, the data cannot be decimated. For example:

- :WAVEform:POINts 1000 – returns time buckets 0, 1, 2, 3, 4, ..., 999.
- :WAVEform:POINts 500 – returns time buckets 0, 2, 4, 6, 8, ..., 998.
- :WAVEform:POINts 250 – returns time buckets 0, 4, 8, 12, 16, ..., 996.
- :WAVEform:POINts 100 – returns time buckets 0, 10, 20, 30, 40, ..., 990.

## Analog Channel Data

### **NORMal Data**

Normal data consists of the last data point (hit) in each time bucket. This data is transmitted over the programming interface in a linear fashion starting with time bucket 0 and going through time bucket n - 1, where n is the number returned by the :WAVEform:POINts? query (see [page 1158](#)). Only the magnitude values of

each data point are transmitted. The first voltage value corresponds to the first time bucket on the left side of the screen and the last value corresponds to the next-to-last time bucket on the right side of the screen. Time buckets without data return 0. The time values for each data point correspond to the position of the data point in the data array. These time values are not transmitted.

### **AVERage Data**

AVERage data consists of the average of the first n hits in a time bucket, where n is the value returned by the :ACQuire:COUNT query (see [page 246](#)). Time buckets that have fewer than n hits return the average of the data they do have. If a time bucket does not have any data in it, it returns 0.

This data is transmitted over the interface linearly, starting with time bucket 0 and proceeding through time bucket n-1, where n is the number returned by the :WAVeform:POINTs? query (see [page 1158](#)). The first value corresponds to a point at the left side of the screen and the last value corresponds to one point away from the right side of the screen. The maximum number of points that can be returned in average mode is 1000 unless ACQuire:COUNT has been set to 1.

### **PEAK Data**

Peak detect display mode is used to detect glitches for time base settings of 500 us/div and slower. In this mode, the oscilloscope can sample more data than it can store and display. So, when peak detect is turned on, the oscilloscope scans through the extra data, picks up the minimum and maximum for each time bucket, then stores the data in an array. Each time bucket contains two data sample.

The array is transmitted over the interface bus linearly, starting with time bucket 0 proceeding through time bucket n-1, where n is the number returned by the :WAVeform:POINTs? query (see [page 1158](#)). In each time bucket, two values are transmitted, first the minimum, followed by the maximum. The first pair of values corresponds to the time bucket at the leftmost side of the screen. The last pair of values corresponds to the time bucket at the far right side of the screen. In :ACQuire:TYPE PEAK mode (see [page 255](#)), the value returned by the :WAVeform:XINCrement query (see [page 1175](#)) should be doubled to find the time difference between the min-max pairs.

### **HRESolution Data**

The high resolution (*smoothing*) mode is used to reduce noise at slower sweep speeds where the digitizer samples faster than needed to fill memory for the displayed time range.

### **Data Conversion**

Word or byte data sent from the oscilloscope must be scaled for useful interpretation. The values used to interpret the data are the X and Y references, X and Y origins, and X and Y increments. These values are read from the waveform preamble. Each channel has its own waveform preamble.

In converting a data value to a voltage value, the following formula is used:

voltage = [(data value - yreference) \* yincrement] + yorigin

If the :WAVEform:FORMAT data format is ASCII (see [page 1157](#)), the data values are converted internally and sent as floating point values separated by commas.

In converting a data value to time, the time value of a data point can be determined by the position of the data point. For example, the fourth data point sent with :WAVEform:XORigin = 16 ns, :WAVEform:XREFerence = 0, and :WAVEform:XINCrement = 2 ns, can be calculated using the following formula:

time = [(data point number - xreference) \* xincrement] + xorigin

This would result in the following calculation for time bucket 3:

time = [(3 - 0) \* 2 ns] + 16 ns = 22 ns

In :ACQuire:TYPE PEAK mode (see [page 255](#)), because data is acquired in max-min pairs, modify the previous time formula to the following:

time=[(data pair number - xreference) \* xincrement \* 2] + xorigin

### **Data Format for Transfer**

There are three formats for transferring waveform data over the interface: BYTE, WORD and ASCII (see "[:WAVEform:FORMAT](#)" on page 1157). BYTE, WORD and ASCII formatted waveform records are transmitted using the arbitrary block program data format specified in IEEE 488.2.

When you use the block data format, the ASCII character string "#8<DD...D>" is sent prior to sending the actual data. The 8 indicates how many Ds follow. The Ds are ASCII numbers that indicate how many data bytes follow.

For example, if 1000 points will be transferred, and the WORD format was specified, the block header "#800001000" would be sent. The 8 indicates that eight length bytes follow, and 00001000 indicates that 1000 binary data bytes follow.

Use the :WAVEform:UNSIGNED command (see [page 1173](#)) to control whether data values are sent as unsigned or signed integers. This command can be used to match the instrument's internal data type to the data type used by the programming language. This command has no effect if the data format is ASCII.

#### **Data Format for Transfer - ASCII format**

The ASCII format (see "[:WAVEform:FORMAT](#)" on page 1157) provides access to the waveform data as real Y-axis values without using Y origin, Y reference, and Y increment to convert the binary data. Values are transferred as ASCII digits in floating point format separated by commas. In ASCII format, holes are represented by the value 9.9e+37. The setting of :WAVEform:BYTeorder (see [page 1153](#)) and :WAVEform:UNSIGNED (see [page 1173](#)) have no effect when the format is ASCII.

#### **Data Format for Transfer - WORD format**

WORD format (see "[:WAVeform:FORMat](#)" on page 1157) provides 16-bit access to the waveform data. In the WORD format, the number of data bytes is twice the number of data points. The number of data points is the value returned by the :WAVeform:POINts? query (see [page 1158](#)). If the data intrinsically has less than 16 bits of resolution, the data is left-shifted to provide 16 bits of resolution and the least significant bits are set to 0. Currently, the greatest intrinsic resolution of any data is 12 bits, so at least the lowest 4 bits of data will be 0. If there is a hole in the data, the hole is represented by a 16 bit value equal to 0.

Use :WAVeform:BYTeorder (see [page 1153](#)) to determine if the least significant byte or most significant byte is to be transferred first. The :BYTeorder command can be used to alter the transmit sequence to match the storage sequence of an integer in the programming language being used.

#### Data Format for Transfer - BYTE format

The BYTE format (see "[:WAVeform:FORMat](#)" on page 1157 ) allows 8-bit access to the waveform data. If the data intrinsically has more than 8 bits of resolution (averaged data), the data is right-shifted (truncated) to fit into 8 bits. If there is a hole in the data, the hole is represented by a value of 0. The BYTE-formatted data transfers over the programming interface faster than ASCII or WORD-formatted data, because in ASCII format, as many as 13 bytes per point are transferred, in BYTE format one byte per point is transferred, and in WORD format two bytes per point are transferred.

The :WAVeform:BYTeorder command (see [page 1153](#)) has no effect when the data format is BYTE.

#### Digital Channel Data (MSO models only)

The waveform record for digital channels is similar to that of analog channels. The main difference is that the data points represent either DIGital0,..,7 (POD1), DIGital8,..,15 (POD2), or any grouping of digital channels (BUS1 or BUS2).

For digital channels, :WAVeform:UNSIGNED (see [page 1173](#)) must be set to ON.

#### Digital Channel POD Data Format

Data for digital channels is only available in groups of 8 bits (Pod1 = D0 - D7, Pod2 = D8 - D15). The bytes are organized as:

:WAVeform:SOURce	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POD1	D7	D6	D5	D4	D3	D2	D1	D0
POD2	D15	D14	D13	D12	D11	D10	D9	D8

If the :WAVeform:FORMat is WORD (see [page 1157](#)) is WORD, every other data byte will be 0. The setting of :WAVeform:BYTeorder (see [page 1153](#)) controls which byte is 0.

If a digital channel is not displayed, its bit value in the pod data byte is not defined.

### Digital Channel BUS Data Format

Digital channel BUS definitions can include any or all of the digital channels. Therefore, data is always returned as 16-bit values. :BUS commands (see [page 257](#)) are used to select the digital channels for a bus.

### Reporting the Setup

The following is a sample response from the :WAVeform? query. In this case, the query was issued following a \*RST command.

```
:WAV:UNS 1;VIEW MAIN;BYT MSBF;FORM BYTE;POIN +1000;SOUR CHAN1;SOUR:SUBS  
NONE
```

## :WAVeform:BYTeorder



(see [page 1334](#))

**Command Syntax**    `:WAVeform:BYTeorder <value>`  
`<value> ::= {LSBFFirst | MSBFFirst}`

The :WAVeform:BYTeorder command sets the output sequence of the WORD data. The parameter MSBFFirst sets the most significant byte to be transmitted first. The parameter LSBFirst sets the least significant byte to be transmitted first. This command affects the transmitting sequence only when :WAVeform:FORMAT WORD is selected. The default setting is LSBFirst.

**Query Syntax**    `:WAVeform:BYTeorder?`

The :WAVeform:BYTeorder query returns the current output sequence.

**Return Format**    `<value><NL>`  
`<value> ::= {LSBF | MSBF}`

**See Also**

- ["Introduction to :WAVeform Commands"](#) on page 1147
- [":WAVeform:DATA"](#) on page 1155
- [":WAVeform:FORMAT"](#) on page 1157
- [":WAVeform:PREamble"](#) on page 1162

**Example Code**

- ["Example Code"](#) on page 1168
- ["Example Code"](#) on page 1163

**:WAVeform:COUNT**(see [page 1334](#))**Query Syntax**    `:WAVeform:COUNT?`

The `:WAVeform:COUNT?` query returns the count used to acquire the current waveform. This may differ from current values if the unit has been stopped and its configuration modified. For all acquisition types except average, this value is 1.

**Return Format**    `<count_argument><NL>`

`<count_argument>` ::= an integer from 1 to 65536 in NR1 format

**See Also**

- ["Introduction to :WAVeform Commands"](#) on page 1147
- [":ACQuire:COUNT"](#) on page 246
- [":ACQuire:TYPE"](#) on page 255

## :WAVeform:DATA



(see [page 1334](#))

### Query Syntax :WAVeform:DATA?

The :WAVeform:DATA query returns the binary block of sampled data points transmitted using the IEEE 488.2 arbitrary block data format. The binary data is formatted according to the settings of the :WAVeform:UNSIGNED, :WAVeform:BYTeorder, :WAVeform:FORMAT, and :WAVeform:SOURce commands. The number of points returned is controlled by the :WAVeform:POINTS command.

In BYTE or WORD waveform formats, these data values have special meaning:

- 0x00 or 0x0000 – Hole. Holes are locations where data has not yet been acquired.
- Another situation where there can be zeros in the data, incorrectly, is when programming over telnet port 5024. Port 5024 provides a command prompt and is intended for ASCII transfers. Use telnet port 5025 instead.
- 0x01 or 0x0001 – Clipped low. These are locations where the waveform is clipped at the bottom of the oscilloscope display.
- 0xFF or 0xFFFF – Clipped high. These are locations where the waveform is clipped at the top of the oscilloscope display.

### Return Format <binary block data><NL>

### See Also

- For a more detailed description of the data returned for different acquisition types, see: "[Introduction to :WAVeform Commands](#)" on page 1147
- "[:WAVeform:UNSIGNED](#)" on page 1173
- "[:WAVeform:BYTeorder](#)" on page 1153
- "[:WAVeform:FORMAT](#)" on page 1157
- "[:WAVeform:POINTS](#)" on page 1158
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:SOURce](#)" on page 1167
- "[:WAVeform:TYPE](#)" on page 1172

### Example Code

```

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString ":WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
'
'     <header><waveform_data><NL>
'
' Where:
'     <header> = #800001000 (This is an example header)

```

```

' The "#8" may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block. The
' size can vary depending on the number of points acquired for the
' waveform. You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
' Unsigned integer bytes.
For lngI = 0 To UBound(varQueryResult) -
    Step (UBound(varQueryResult) / 20) - ' 20 points.
    If intBytesPerData = 2 Then
        lngDataValue = varQueryResult(lngI) * 256 -
            + varQueryResult(lngI + 1) ' 16-bit value.
    Else
        lngDataValue = varQueryResult(lngI) ' 8-bit value.
    End If
    strOutput = strOutput + "Data point " + _
        CStr(lngI / intBytesPerData) + ", " + _
        FormatNumber((lngDataValue - lngYReference) -
            * sngYIncrement + sngYOrigin) + " V, " + _
        FormatNumber(((lngI / intBytesPerData - lngXReference) -
            * sngXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :WAVeform:FORMAT



(see [page 1334](#))

### Command Syntax

```
:WAVeform:FORMAT <value>
<value> ::= {WORD | BYTE | ASCii}
```

The :WAVeform:FORMAT command sets the data transmission mode for waveform data points. This command controls how the data is formatted when sent from the oscilloscope.

- ASCii formatted data converts the internal integer data values to real Y-axis values. Values are transferred as ASCii digits in floating point notation, separated by commas.  
ASCii formatted data is transferred ASCii text.
- WORD formatted data transfers 16-bit data as two bytes. The :WAVeform:BYTeorder command can be used to specify whether the upper or lower byte is transmitted first. The default (no command sent) is that the upper byte transmitted first.
- BYTE formatted data is transferred as 8-bit bytes.

When the :WAVeform:SOURce is the serial decode bus (SBUS1 or SBUS2), ASCii is the only waveform format allowed.

When the :WAVeform:SOURce is one of the digital channel buses (BUS1 or BUS2), ASCii and WORD are the only waveform formats allowed.

### Query Syntax

```
:WAVeform:FORMAT?
```

The :WAVeform:FORMAT query returns the current output format for the transfer of waveform data.

### Return Format

```
<value><NL>
<value> ::= {WORD | BYTE | ASC}
```

### See Also

- "[Introduction to :WAVeform Commands](#)" on page 1147
- "[":WAVeform:BYTeorder](#)" on page 1153
- "[":WAVeform:SOURce](#)" on page 1167
- "[":WAVeform:DATA](#)" on page 1155
- "[":WAVeform:PREamble](#)" on page 1162

### Example Code

- "[Example Code](#)" on page 1168

## :WAVEform:POINTs

**C** (see [page 1334](#))

**Command Syntax**

```
:WAVEform:POINTs <# points>

<# points> ::= {100 | 250 | 500 | 1000 | <points mode>}
                if waveform points mode is NORMAl

<# points> ::= {100 | 250 | 500 | 1000 | 2000 | 5000 | 10000 | 20000
                | 50000 | 100000 | 200000 | 500000 | 1000000 | 2000000
                | 4000000 | 8000000 | <points mode>}
                if waveform points mode is MAXimum or RAW

<points mode> ::= {NORMAl | MAXimum | RAW}
```

### NOTE

The <points\_mode> option is deprecated. Use the :WAVEform:POINTs:MODE command instead.

The :WAVEform:POINTs command sets the number of waveform points to be transferred with the :WAVEform:DATA? query. This value represents the points contained in the waveform selected with the :WAVEform:SOURce command.

For the analog or digital sources, the records that can be transferred depend on the waveform points mode. The maximum number of points returned for math (function) waveforms is determined by the NORMAl waveform points mode. See the :WAVEform:POINTs:MODE command (see [page 1160](#)) for more information.

Only data visible on the display will be returned.

When the :WAVEform:SOURce is the serial decode bus (SBUS1 or SBUS2), this command is ignored, and all available serial decode bus data is returned.

**Query Syntax**

:WAVEform:POINTs?

The :WAVEform:POINTs query returns the number of waveform points to be transferred when using the :WAVEform:DATA? query. Setting the points mode will affect what data is transferred (see the :WAVEform:POINTs:MODE command (see [page 1160](#)) for more information).

When the :WAVEform:SOURce is the serial decode bus (SBUS1 or SBUS2), this query returns the number of messages that were decoded.

**Return Format**

```
<# points><NL>

<# points> ::= {100 | 250 | 500 | 1000 | <maximum # points>}
                if waveform points mode is NORMAl

<# points> ::= {100 | 250 | 500 | 1000 | 2000 | 5000 | 10000 | 20000
                | 50000 | 100000 | 200000 | 500000 | 1000000 | 2000000
                | 4000000 | 8000000 | <maximum # points>}
                if waveform points mode is MAXimum or RAW
```

**NOTE**

If a full screen of data is not displayed, the number of points returned will not be 1000 or an even divisor of it.

---

**See Also**

- "[Introduction to :WAVeform Commands](#)" on page 1147
- "[:ACQuire:POINts](#)" on page 248
- "[:WAVeform:DATA](#)" on page 1155
- "[:WAVeform:SOURce](#)" on page 1167
- "[:WAVeform:VIEW](#)" on page 1174
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:POINts:MODE](#)" on page 1160

**Example Code**

```
' WAVE_POINTS - Specifies the number of points to be transferred  
' using the ":WAVEFORM:DATA?" query.  
myScope.WriteString ":WAVEFORM:POINTS 1000"
```

See complete example programs at: [Chapter 42](#), "Programming Examples," starting on page 1343

## :WAVEform:POINTS:MODE

**N** (see [page 1334](#))

**Command Syntax**    `:WAVEform:POINTS:MODE <points_mode>`  
`<points_mode> ::= {NORMAl | MAXimum | RAW}`

The :WAVEform:POINTS:MODE command sets the data record to be transferred with the :WAVEform:DATA? query.

For the analog or digital sources, there are two different records that can be transferred:

- The first is the raw acquisition record. The maximum number of points available in this record is returned by the :ACQuire:POINTS? query. The raw acquisition record can only be retrieved from the analog or digital sources.
- The second is referred to as the *measurement record* and is a 62,500-point (maximum) representation of the raw acquisition record. The measurement record can be retrieved from any source.

If the <points\_mode> is NORMAl the measurement record is retrieved.

If the <points\_mode> is RAW, the raw acquisition record is used. Under some conditions, this data record is unavailable.

If the <points\_mode> is MAXimum, whichever record contains the maximum amount of points is used. Usually, this is the raw acquisition record. But, the measurement record may have more data. If data is being retrieved as the oscilloscope is stopped and as the data displayed is changing, the data being retrieved can switch between the measurement and raw acquisition records.

- Considerations for MAXimum or RAW data retrieval**
- The instrument must be stopped (see the :STOP command (see [page 240](#)) or the :DIGItize command (see [page 215](#)) in the root subsystem) in order to return more than the *measurement record*.
  - :TIMEbase:MODE must be set to MAIN.
  - :ACQuire:TYPE must be set to NORMAl, AVERage, or HRESolution.
  - MAXimum or RAW will allow up to 4,000,000 points to be returned. The number of points returned will vary as the instrument's configuration is changed. Use the :WAVEform:POINTS? MAXimum query to determine the maximum number of points that can be retrieved at the current settings.

**Query Syntax**    `:WAVEform:POINTS:MODE?`

The :WAVEform:POINTS:MODE? query returns the current points mode. Setting the points mode will affect what data is transferred. See the discussion above.

**Return Format**    `<points_mode><NL>`  
`<points_mode> ::= {NORMAl | MAXimum | RAW}`

**See Also**    • "Introduction to :WAVEform Commands" on page 1147

- [":WAveform:DATA" on page 1155](#)
- [":ACQuire:POINts" on page 248](#)
- [":WAveform:VIEW" on page 1174](#)
- [":WAveform:PREamble" on page 1162](#)
- [":WAveform:POINts" on page 1158](#)
- [":TIMEbase:MODE" on page 1037](#)
- [":ACQuire:TYPE" on page 255](#)
- [":ACQuire:COUNT" on page 246](#)

## :WAVEform:PREamble



(see page 1334)

**Query Syntax**    `:WAVEform:PREamble?`

The `:WAVEform:PREamble` query requests the preamble information for the selected waveform source. The preamble data contains information concerning the vertical and horizontal scaling of the data of the corresponding channel.

**Return Format**    `<preamble_block><NL>`

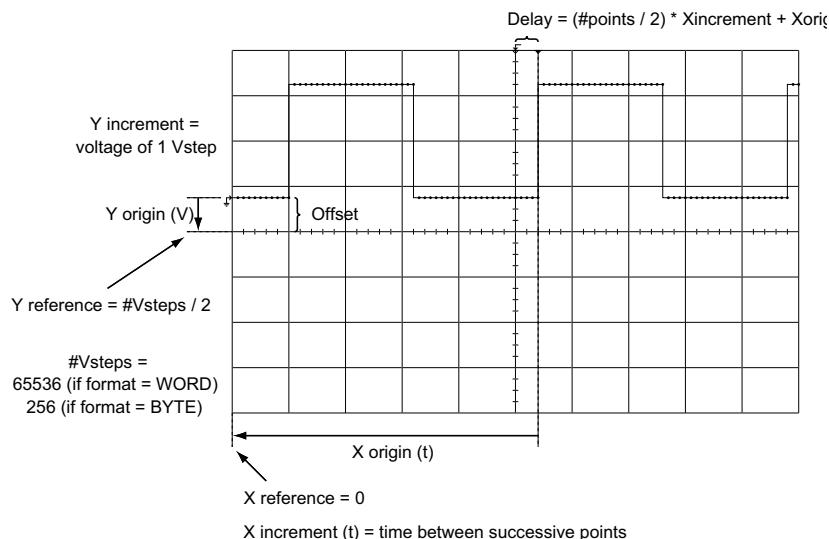
```

<preamble_block> ::= <format 16-bit NR1>,
                    <type 16-bit NR1>,
                    <points 32-bit NR1>,
                    <count 32-bit NR1>,
                    <xincrement 64-bit floating point NR3>,
                    <xorigin 64-bit floating point NR3>,
                    <xreference 32-bit NR1>,
                    <yincrement 32-bit floating point NR3>,
                    <yorigin 32-bit floating point NR3>,
                    <yreference 32-bit NR1>

<format> ::= 0 for BYTE format, 1 for WORD format, 4 for ASCii format;
            an integer in NR1 format (format set by :WAVEform:FORMAT).

<type> ::= 3 for HRESolution type, 2 for AVERage type, 0 for NORMAl
            type, 1 for PEAK detect type; an integer in NR1 format
            (type set by :ACQuire:TYPE).

<count> ::= Average count or 1 if PEAK or NORMAl; an integer in NR1
            format (count set by :ACQuire:COUNT).
  
```



**See Also**

- "Introduction to :WAVEform Commands" on page 1147
- ":ACQuire:COUNT" on page 246

- "[:ACQuire:POINTs](#)" on page 248
- "[:ACQuire:TYPE](#)" on page 255
- "[:DIGitize](#)" on page 215
- "[:WAVeform:COUNT](#)" on page 1154
- "[:WAVeform:DATA](#)" on page 1155
- "[:WAVeform:FORMAT](#)" on page 1157
- "[:WAVeform:POINTs](#)" on page 1158
- "[:WAVeform:TYPE](#)" on page 1172
- "[:WAVeform:XINCrement](#)" on page 1175
- "[:WAVeform:XORigin](#)" on page 1176
- "[:WAVeform:XREFERENCE](#)" on page 1177
- "[:WAVeform:YINCrement](#)" on page 1178
- "[:WAVeform:YORigin](#)" on page 1179
- "[:WAVeform:YREFERENCE](#)" on page 1180

**Example Code**

```

' GET_PREAMBLE - The preamble block contains all of the current
' WAVEFORM settings. It is returned in the form <preamble_block><NL>
' where <preamble_block> is:
'   FORMAT      : int16 - 0 = BYTE, 1 = WORD, 4 = ASCII.
'   TYPE        : int16 - 0 = NORM, 1 = PEAK, 2 = AVER, 3 = HRES
'   POINTS     : int32 - number of data points transferred.
'   COUNT       : int32 - 1 and is always 1.
'   XINCREMENT : float64 - time difference between data points.
'   XORIGIN    : float64 - always the first data point in memory.
'   XREFERENCE : int32 - specifies the data point associated with
'                      x-origin.
'   YINCREMENT : float32 - voltage diff between data points.
'   YORIGIN    : float32 - value is the voltage at center screen.
'   YREFERENCE : int32 - specifies the data point where y-origin
'                      occurs.

Dim Preamble()
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim sngYOrigin As Single
Dim lngYReference As Long
Dim strOutput As String

myScope.WriteString ":WAVEFORM:PREAMBLE?"      ' Query for the preamble.
Preamble() = myScope.ReadList      ' Read preamble information.
intFormat = Preamble(0)
intType = Preamble(1)
lngPoints = Preamble(2)

```

```
    lngCount = Preamble(3)
    dblXIncrement = Preamble(4)
    dblXOrigin = Preamble(5)
    lngXReference = Preamble(6)
    sngYIncrement = Preamble(7)
    sngYOrigin = Preamble(8)
    lngYReference = Preamble(9)
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :WAVeform:SEGmented:COUNt

**N** (see [page 1334](#))

**Query Syntax** :WAVeform:SEGmented:COUNt?

**NOTE** This command is available when the segmented memory option (Option SGM) is enabled.

The :WAVeform:SEGmented:COUNt query returns the number of memory segments in the acquired data. You can use the :WAVeform:SEGmented:COUNt? query while segments are being acquired (although :DIGitize blocks subsequent queries until the full segmented acquisition is complete).

The segmented memory acquisition mode is enabled with the :ACQuire:MODE command. The number of segments to acquire is set using the :ACQuire:SEGmented:COUNt command, and data is acquired using the :DIGITIZE, :SINGle, or :RUN commands.

**Return Format** <count> ::= an integer from 2 to 1000 in NR1 format (count set by :ACQuire:SEGmented:COUNt).

- See Also**
- [":ACQuire:MODE"](#) on page 247
  - [":ACQuire:SEGmented:COUNt"](#) on page 250
  - [":DIGITIZE"](#) on page 215
  - [":SINGle"](#) on page 238
  - [":RUN"](#) on page 236
  - ["Introduction to :WAVeform Commands"](#) on page 1147

**Example Code**

- ["Example Code"](#) on page 251

**:WAVeform:SEGmented:TTAG****N** (see [page 1334](#))**Query Syntax**    `:WAVeform:SEGmented:TTAG?`**NOTE** This command is available when the segmented memory option (Option SGM) is enabled.

The `:WAVeform:SEGmented:TTAG?` query returns the time tag of the currently selected segmented memory index. The index is selected using the `:ACQuire:SEGmented:INDex` command.

**Return Format**    `<time_tag> ::= in NR3 format`

- See Also**
- " [":ACQuire:SEGmented:INDex](#)" on page 251
  - " [Introduction to :WAVeform Commands](#)" on page 1147

**Example Code**

- " [Example Code](#)" on page 251

## :WAVeform:SOURce

**C** (see [page 1334](#))

### Command Syntax :WAVeform:SOURce <source>

```
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m> | FFT | WMEMory<r>
              | SBUS{1 | 2}} for DSO models

<source> ::= {CHANnel<n> | POD{1 | 2} | BUS{1 | 2} | FUNCTion<m>
              | MATH<m> | FFT | WMEMory<r> | SBUS{1 | 2}}
              for MSO models

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :WAVeform:SOURce command selects the analog channel, function, digital pod, digital bus, reference waveform, or serial decode bus to be used as the source for the :WAVeform commands.

Function capabilities include add, subtract, multiply, integrate, differentiate, and FFT (Fast Fourier Transform) operations.

When the :WAVeform:SOURce is the serial decode bus (SBUS1 or SBUS2), ASCII is the only waveform format allowed, and the :WAVeform:DATA? query returns a string with timestamps and associated bus decode information.

With MSO oscilloscope models, you can choose a POD or BUS as the waveform source. There are some differences between POD and BUS when formatting and getting data from the oscilloscope:

- When POD1 or POD2 is selected as the waveform source, you can choose the BYTE, WORD, or ASCII formats (see "[:WAVeform:FORMat](#)" on page 1157).

When the WORD format is chosen, every other data byte will be 0. The setting of :WAVeform:BYTeorder controls which byte is 0.

When the ASCII format is chosen, the :WAVeform:DATA? query returns a string with unsigned decimal values separated by commas.

- When BUS1 or BUS2 is selected as the waveform source, you can choose the WORD or ASCII formats (but not BYTE because bus values are always returned as 16-bit values).

When the ASCII format is chosen, the :WAVeform:DATA? query returns a string with hexadecimal bus values, for example: 0x1938,0xff38,...

### Query Syntax :WAVeform:SOURce?

The :WAVeform:SOURce? query returns the currently selected source for the WAVeform commands.

**NOTE**

MATH<m> is an alias for FUNCtion<m>. The :WAVeform:SOURce? query returns FUNC<m> if the source is FUNCtion<m> or MATH<m>.

Return Format	<pre>&lt;source&gt;&lt;NL&gt; &lt;source&gt; ::= {CHAN&lt;n&gt;   FUNC&lt;m&gt;   WMEM&lt;r&gt;   SBUS{1   2}} for DSO models &lt;source&gt; ::= {CHAN&lt;n&gt;   POD{1   2}   BUS{1   2}   FUNC&lt;m&gt;                  WMEM&lt;r&gt;   SBUS{1   2}} for MSO models &lt;n&gt; ::= 1 to (# analog channels) in NR1 format &lt;m&gt; ::= 1 to (# math functions) in NR1 format &lt;r&gt; ::= 1 to (# ref waveforms) in NR1 format</pre>
See Also	<ul style="list-style-type: none"> <li>• "<a href="#">Introduction to :WAVeform Commands</a>" on page 1147</li> <li>• "<a href="#">:DIGitize</a>" on page 215</li> <li>• "<a href="#">:WAVeform:FORMat</a>" on page 1157</li> <li>• "<a href="#">:WAVeform:BYTeorder</a>" on page 1153</li> <li>• "<a href="#">:WAVeform:DATA</a>" on page 1155</li> <li>• "<a href="#">:WAVeform:PREamble</a>" on page 1162</li> </ul>
Example Code	<pre>' WAVEFORM_DATA - To obtain waveform data, you must specify the ' WAVEFORM parameters for the waveform data prior to sending the ' ":WAVEFORM:DATA?" query. Once these parameters have been sent, ' the waveform data and the preamble can be read. ' ' WAVE_SOURCE - Selects the channel to be used as the source for ' the waveform commands. myScope.WriteString ":WAVEFORM:SOURCE CHAN1"  ' WAVE_POINTS - Specifies the number of points to be transferred ' using the ":WAVEFORM:DATA?" query. myScope.WriteString ":WAVEFORM:POINTS 1000"  ' WAVE_FORMAT - Sets the data transmission mode for the waveform ' data output. This command controls whether data is formatted in ' a word or byte format when sent from the oscilloscope. Dim lngVSteps As Long Dim intBytesPerData As Integer  ' Data in range 0 to 65535. myScope.WriteString ":WAVEFORM:FORMAT WORD" lngVSteps = 65536 intBytesPerData = 2  ' Data in range 0 to 255. 'myScope.WriteString ":WAVEFORM:FORMAT BYTE" 'lngVSteps = 256 'intBytesPerData = 1  ' GET_PREAMBLE - The preamble block contains all of the current</pre>

```

' WAVEFORM settings. It is returned in the form <preamble_block><NL>
' where <preamble_block> is:
'   FORMAT      : int16 - 0 = BYTE, 1 = WORD, 4 = ASCII.
'   TYPE        : int16 - 0 = NORMAL, 1 = PEAK DETECT, 2 = AVERAGE
'   POINTS     : int32 - number of data points transferred.
'   COUNT       : int32 - 1 and is always 1.
'   XINCREMENT  : float64 - time difference between data points.
'   XORIGIN     : float64 - always the first data point in memory.
'   XREFERENCE  : int32 - specifies the data point associated with
'                     x-origin.
'   YINCREMENT  : float32 - voltage diff between data points.
'   YORIGIN     : float32 - value is the voltage at center screen.
'   YREFERENCE  : int32 - specifies the data point where y-origin
'                     occurs.

Dim Preamble()
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim sngYOrigin As Single
Dim lngYReference As Long
Dim strOutput As String

myScope.WriteString ":WAVEFORM:PREAMBLE?" ' Query for the preamble.
Preamble() = myScope.ReadList ' Read preamble information.
intFormat = Preamble(0)
intType = Preamble(1)
lngPoints = Preamble(2)
lngCount = Preamble(3)
dblXIncrement = Preamble(4)
dblXOrigin = Preamble(5)
lngXReference = Preamble(6)
sngYIncrement = Preamble(7)
sngYOrigin = Preamble(8)
lngYReference = Preamble(9)
strOutput = ""
strOutput = strOutput + "Format = " + CStr(intFormat) + vbCrLf
strOutput = strOutput + "Type = " + CStr(intType) + vbCrLf
strOutput = strOutput + "Points = " + CStr(lngPoints) + vbCrLf
strOutput = strOutput + "Count = " + CStr(lngCount) + vbCrLf
strOutput = strOutput + "X increment = " +
'           FormatNumber(dblXIncrement * 1000000) + " us" + vbCrLf
strOutput = strOutput + "X origin = " +
'           FormatNumber(dblXOrigin * 1000000) + " us" + vbCrLf
strOutput = strOutput + "X reference = " +
'           CStr(lngXReference) + vbCrLf
strOutput = strOutput + "Y increment = " +
'           FormatNumber(sngYIncrement * 1000) + " mV" + vbCrLf
strOutput = strOutput + "Y origin = " +
'           FormatNumber(sngYOrigin) + " V" + vbCrLf
strOutput = strOutput + "Y reference = " +
'           CStr(lngYReference) + vbCrLf
strOutput = strOutput + "Volts/Div = " +
'
```

```

        FormatNumber(lngVSteps * sngYIncrement / 8) + _
        " V" + vbCrLf
strOutput = strOutput + "Offset = " + _
        FormatNumber((lngVSteps / 2 - lngYReference) * _
        sngYIncrement + sngYOrigin) + " V" + vbCrLf
strOutput = strOutput + "Sec/Div = " + _
        FormatNumber(lngPoints * dblXIncrement / 10 * _
        1000000) + " us" + vbCrLf
strOutput = strOutput + "Delay = " + _
        FormatNumber(((lngPoints / 2 - lngXReference) * _
        dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString ":WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
'
' <header><waveform_data><NL>
'
' Where:
'   <header> = #800001000 (This is an example header)
' The "#8" may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block. The
' size can vary depending on the number of points acquired for the
' waveform. You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

' Unsigned integer bytes.
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)

For lngI = 0 To UBound(varQueryResult) -
    Step (UBound(varQueryResult) / 20)      ' 20 points.
    If intBytesPerData = 2 Then
        lngDataValue = varQueryResult(lngI) * 256 -
            + varQueryResult(lngI + 1)      ' 16-bit value.
    Else
        lngDataValue = varQueryResult(lngI)      ' 8-bit value.
    End If
    strOutput = strOutput + "Data point " + _
        CStr(lngI / intBytesPerData) + ", " + _
        FormatNumber((lngDataValue - lngYReference) _ 
        * sngYIncrement + sngYOrigin) + " V, " + _
        FormatNumber(((lngI / intBytesPerData - lngXReference) _ 
        * sngXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :WAVeform:SOURce:SUBSource



(see [page 1334](#))

### Command Syntax

```
:WAVeform:SOURce:SUBSource <subsource>
<subsource> ::= {{SUB0 | RX | MOSI | FAST}
                  | {SUB1 | TX | MISO | SLOW}}
```

If the :WAVeform:SOURce is SBUS<n> (serial decode), more than one data set may be available, and this command lets you choose from the available data sets.

When using UART serial decode, this option lets you get "TX" data. (TX is an alias for SUB1.) The default, SUB0, specifies "RX" data. (RX is an alias for SUB0.)

When using SPI serial decode, this option lets you get "MISO" data. (MISO is an alias for SUB1.) The default, SUB0, specifies "MOSI" data. (MOSI is an alias for SUB0.)

When using SENT serial decode, this option lets you get "SLOW" data. (SLOW is an alias for SUB1.) The default, SUB0, specifies "FAST" data. (FAST is an alias for SUB0.)

If the :WAVeform:SOURce is not SBUS, or the :SBUS<n>:MODE is not UART, SPI, or SENT, the only valid subsource is SUB0.

### Query Syntax

```
:WAVeform:SOURce:SUBSource?
```

The :WAVeform:SOURce:SUBSource? query returns the current waveform subsource setting.

### Return Format

```
<subsource><NL>
<subsource> ::= {SUB0 | SUB1}
```

### See Also

- ["Introduction to :WAVeform Commands" on page 1147](#)
- [":WAVeform:SOURce" on page 1167](#)

**:WAVeform:TYPE**(see [page 1334](#))**Query Syntax**    `:WAVeform:TYPE?`

The `:WAVeform:TYPE?` query returns the acquisition mode associated with the currently selected waveform. The acquisition mode is set by the `:ACQuire:TYPE` command.

**Return Format**    `<mode><NL>`

`<mode> ::= {NORM | PEAK | AVER | HRES}`

**NOTE**

If the `:WAVeform:SOURce` is POD1, POD2, or SBUS1, SBUS2, the type is always NORM.

**See Also**

- "[Introduction to :WAVeform Commands](#)" on page 1147
- "[":ACQuire:TYPE](#)" on page 255
- "[":WAVeform:DATA](#)" on page 1155
- "[":WAVeform:PREamble](#)" on page 1162
- "[":WAVeform:SOURce](#)" on page 1167

## :WAVeform:UNSIGNED



(see [page 1334](#))

**Command Syntax**    `:WAVeform:UNSIGNED <unsigned>`  
`<unsigned> ::= {{0 | OFF} | {1 | ON}}`

The :WAVeform:UNSIGNED command turns unsigned mode on or off for the currently selected waveform. Use the WAVeform:UNSIGNED command to control whether data values are sent as unsigned or signed integers. This command can be used to match the instrument's internal data type to the data type used by the programming language. This command has no effect if the data format is ASCII.

If :WAVeform:SOURce is set to POD1, POD2, BUS1, or BUS2, WAVeform:UNSIGNED must be set to ON.

**Query Syntax**    `:WAVeform:UNSIGNED?`

The :WAVeform:UNSIGNED? query returns the status of unsigned mode for the currently selected waveform.

**Return Format**    `<unsigned><NL>`  
`<unsigned> ::= {0 | 1}`

**See Also**

- "Introduction to :WAVeform Commands" on page 1147
- "[:WAVeform:SOURce](#)" on page 1167

**:WAveform:VIEW**(see [page 1334](#))

**Command Syntax**    `:WAveform:VIEW <view>`  
                      `<view> ::= {MAIN}`

The :WAveform:VIEW command sets the view setting associated with the currently selected waveform. Currently, the only legal value for the view setting is MAIN.

**Query Syntax**    `:WAveform:VIEW?`

The :WAveform:VIEW? query returns the view setting associated with the currently selected waveform.

**Return Format**    `<view><NL>`  
                      `<view> ::= {MAIN}`

**See Also**

- "Introduction to :WAveform Commands" on page 1147
- ":WAveform:POINts" on page 1158

## :WAVeform:XINCrement

(see [page 1334](#))

**Query Syntax** :WAVeform:XINCrement?

The :WAVeform:XINCrement? query returns the x-increment value for the currently specified source. This value is the time difference between consecutive data points in seconds.

**Return Format** <value><NL>

<value> ::= x-increment in the current preamble in 64-bit floating point NR3 format

**See Also** • ["Introduction to :WAVeform Commands" on page 1147](#)  
• [":WAVeform:PREamble" on page 1162](#)

**Example Code** • ["Example Code" on page 1163](#)

**:WAVeform:XORigin**(see [page 1334](#))**Query Syntax**    `:WAVeform:XORigin?`

The `:WAVeform:XORigin?` query returns the x-origin value for the currently specified source. XORigin is the X-axis value of the data point specified by the `:WAVeform:XREFerence` value. In this product, that is always the X-axis value of the first data point (`XREFerence = 0`).

**Return Format**    `<value><NL>`

`<value>` ::= x-origin value in the current preamble in 64-bit floating point NR3 format

**See Also**

- "Introduction to [:WAVeform Commands](#)" on page 1147
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:XREFerence](#)" on page 1177

**Example Code**

- "[Example Code](#)" on page 1163

**:WAVeform:XREFerence**(see [page 1334](#))**Query Syntax**    `:WAVeform:XREFerence?`

The `:WAVeform:XREFerence?` query returns the x-reference value for the currently specified source. This value specifies the index of the data point associated with the x-origin data value. In this product, the x-reference point is the first point displayed and XREFerence is always 0.

**Return Format**    `<value><NL>`

`<value> ::= x-reference value = 0 in 32-bit NR1 format`

**See Also**

- "Introduction to [:WAVeform Commands](#)" on page 1147
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:XORigin](#)" on page 1176

**Example Code**

- "[Example Code](#)" on page 1163

**:WAVeform:YINCrement**(see [page 1334](#))**Query Syntax**    `:WAVeform:YINCrement?`

The `:WAVeform:YINCrement?` query returns the y-increment value in volts for the currently specified source. This value is the voltage difference between consecutive data values. The y-increment for digital waveforms is always "1".

**Return Format**    `<value><NL>`

`<value>` ::= y-increment value in the current preamble in 32-bit floating point NR3 format

**See Also**

- ["Introduction to :WAVeform Commands" on page 1147](#)
- [":WAVeform:PREamble" on page 1162](#)

**Example Code**

- ["Example Code" on page 1163](#)

**:WAVeform:YORigin**(see [page 1334](#))**Query Syntax**    `:WAVeform:YORigin?`

The `:WAVeform:YORigin?` query returns the y-origin value for the currently specified source. This value is the Y-axis value of the data value specified by the `:WAVeform:YREFerence` value. For this product, this is the Y-axis value of the center of the screen.

**Return Format**    `<value><NL>`

`<value>` ::= y-origin in the current preamble in 32-bit floating point NR3 format

**See Also**

- "Introduction to [:WAVeform Commands](#)" on page 1147
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:YREFerence](#)" on page 1180

**Example Code**

- "[Example Code](#)" on page 1163

**:WAVeform:YREFerence**(see [page 1334](#))**Query Syntax**    `:WAVeform:YREFerence?`

The `:WAVeform:YREFerence?` query returns the y-reference value for the currently specified source. This value specifies the data point value where the y-origin occurs. In this product, this is the data point value of the center of the screen. It is undefined if the format is ASCII.

**Return Format**    `<value><NL>`

`<value>` ::= y-reference value in the current preamble in 32-bit NR1 format

**See Also**

- "Introduction to [:WAVeform Commands](#)" on page 1147
- "[:WAVeform:PREamble](#)" on page 1162
- "[:WAVeform:YORigin](#)" on page 1179

**Example Code**

- "[Example Code](#)" on page 1163

## 35 :WGEN<w> Commands

When the built-in waveform generator is licensed (Option WGN), you can use it to output sine, square, ramp, pulse, DC, noise, sine cardinal, exponential rise, exponential fall, cardiac, and gaussian pulse waveforms. The :WGEN<w> commands are used to select the waveform function and parameters. See "[Introduction to :WGEN<w> Commands](#)" on page 1184.

**Table 155**:WGEN<w> Commands Summary

Command	Query	Options and Query Returns
:WGEN<w>:ARBitrary:BYTeorder <order> (see <a href="#">page 1185</a> )	:WGEN<w>:ARBitrary:BYTeorder? (see <a href="#">page 1185</a> )	<order> ::= {MSBFFirst   LSBFirst} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DATA {<binary>   <value>, <value> ...} (see <a href="#">page 1186</a> )	n/a	<binary> ::= floating point values between -1.0 to +1.0 in IEEE 488.2 binary block format <value> ::= floating point values between -1.0 to +1.0 in comma-separated format <w> ::= 1 or 2 in NR1 format
n/a	:WGEN<w>:ARBitrary:DATA:ATTRibute:POINTs? (see <a href="#">page 1187</a> )	<points> ::= number of points in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DATACLEAR (see <a href="#">page 1188</a> )	n/a	<w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:DATADAC {<binary>   <value>, <value> ...} (see <a href="#">page 1189</a> )	n/a	<binary> ::= decimal 16-bit integer values between -512 to +511 in IEEE 488.2 binary block format <value> ::= decimal integer values between -512 to +511 in comma-separated NR1 format <w> ::= 1 or 2 in NR1 format

**Table 155**:WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:ARBitrary:IN Terpolate {{0   OFF}   {1   ON}} (see <a href="#">page 1190</a> )	:WGEN<w>:ARBitrary:IN Terpolate? (see <a href="#">page 1190</a> )	{0   1} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:ARBitrary:ST ORe <source> (see <a href="#">page 1191</a> )	n/a	<source> ::= {CHANnel<n>   WMEMory<r>   FUNCTion<m>   FFT   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FREQuency <frequency> (see <a href="#">page 1192</a> )	:WGEN<w>:FREQuency? (see <a href="#">page 1192</a> )	<frequency> ::= frequency in Hz in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTion <signal> (see <a href="#">page 1193</a> )	:WGEN<w>:FUNCTion? (see <a href="#">page 1196</a> )	<signal> ::= {SINusoid   SQUare   RAMP   PULSe   NOISe   DC   SINC   EXPRIse   EXPFall   CARDiac   GAUSSian   ARBITrary} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTion:PUL Se:WIDTh <width> (see <a href="#">page 1197</a> )	:WGEN<w>:FUNCTion:PUL Se:WIDTh? (see <a href="#">page 1197</a> )	<width> ::= pulse width in seconds in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTion:RAM P:SYMMetry <percent> (see <a href="#">page 1198</a> )	:WGEN<w>:FUNCTion:RAM P:SYMMetry? (see <a href="#">page 1198</a> )	<percent> ::= symmetry percentage from 0% to 100% in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:FUNCTion:SQU are:DCYCle <percent> (see <a href="#">page 1199</a> )	:WGEN<w>:FUNCTion:SQU are:DCYCle? (see <a href="#">page 1199</a> )	<percent> ::= duty cycle percentage from 20% to 80% in NR1 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:MODulation:A M:DEPTh <percent> (see <a href="#">page 1200</a> )	:WGEN<w>:MODulation:A M:DEPTh? (see <a href="#">page 1200</a> )	<percent> ::= AM depth percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:A M:FREQuency <frequency> (see <a href="#">page 1201</a> )	:WGEN<w>:MODulation:A M:FREQuency? (see <a href="#">page 1201</a> )	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format

**Table 155:**:WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:MODulation:F M:DEViation <frequency> (see <a href="#">page 1202</a> )	:WGEN<w>:MODulation:F M:DEViation? (see <a href="#">page 1202</a> )	<frequency> ::= frequency deviation in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F M:FREQuency <frequency> (see <a href="#">page 1203</a> )	:WGEN<w>:MODulation:F M:FREQuency? (see <a href="#">page 1203</a> )	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F SKey:FREQuency <percent> (see <a href="#">page 1204</a> )	:WGEN<w>:MODulation:F SKey:FREQuency? (see <a href="#">page 1204</a> )	<frequency> ::= hop frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F SKey:RATE <rate> (see <a href="#">page 1205</a> )	:WGEN<w>:MODulation:F SKey:RATE? (see <a href="#">page 1205</a> )	<rate> ::= FSK modulation rate in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F UNCtion <shape> (see <a href="#">page 1206</a> )	:WGEN<w>:MODulation:F UNCtion? (see <a href="#">page 1206</a> )	<shape> ::= {SINusoid   SQUare  RAMP} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F UNCtion:RAMP:SYMMetry <percent> (see <a href="#">page 1207</a> )	:WGEN<w>:MODulation:F UNCtion:RAMP:SYMMetry ? (see <a href="#">page 1207</a> )	<percent> ::= symmetry percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:N OISe <percent> (see <a href="#">page 1208</a> )	:WGEN<w>:MODulation:N OISe? (see <a href="#">page 1208</a> )	<percent> ::= 0 to 100 <w> ::= 1 or 2 in NR1 format
:WGEN<w>:MODulation:S TATE {{0   OFF}   {1   ON}} (see <a href="#">page 1209</a> )	:WGEN<w>:MODulation:S TATE? (see <a href="#">page 1209</a> )	{0   1} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:T YPE <type> (see <a href="#">page 1210</a> )	:WGEN<w>:MODulation:T YPE? (see <a href="#">page 1210</a> )	<type> ::= {AM   FM   FSK} <w> ::= 1 in NR1 format
:WGEN<w>:OUTPut {{0   OFF}   {1   ON}} (see <a href="#">page 1212</a> )	:WGEN<w>:OUTPut? (see <a href="#">page 1212</a> )	{0   1} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:LOAD <impedance> (see <a href="#">page 1213</a> )	:WGEN<w>:OUTPut:LOAD? (see <a href="#">page 1213</a> )	<impedance> ::= {ONEMeg   FIFTy} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:MODE <mode> (see <a href="#">page 1214</a> )	:WGEN<w>:OUTPut:MODE? (see <a href="#">page 1214</a> )	<mode> ::= {NORMal   SINGLE}

**Table 155**:WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:OUTPut:POLarity <polarity> (see page 1215)	:WGEN<w>:OUTPut:POLarity? (see page 1215)	<polarity> ::= {NORMAL   INVerted} <w> ::= 1 or 2 in NR1 format
:WGEN<w>:OUTPut:SINGLe (see page 1216)	n/a	n/a
:WGEN<w>:PERiod <period> (see page 1217)	:WGEN<w>:PERiod? (see page 1217)	<period> ::= period in seconds in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:RST (see page 1218)	n/a	<w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage <amplitude> (see page 1219)	:WGEN<w>:VOLTage? (see page 1219)	<amplitude> ::= amplitude in volts in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage:HIGH <high> (see page 1220)	:WGEN<w>:VOLTage:HIGH? (see page 1220)	<high> ::= high-level voltage in volts, in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage:LOW <low> (see page 1221)	:WGEN<w>:VOLTage:LOW? (see page 1221)	<low> ::= low-level voltage in volts, in NR3 format <w> ::= 1 or 2 in NR1 format
:WGEN<w>:VOLTage:OFFSet <offset> (see page 1222)	:WGEN<w>:VOLTage:OFFS et? (see page 1222)	<offset> ::= offset in volts in NR3 format <w> ::= 1 or 2 in NR1 format

**Introduction to :WGEN<w> Commands** The :WGEN<w> subsystem provides commands to select the waveform generator function and parameters.

In the :WGEN<w> commands, the <w> can be 1 or 2, and :WGEN is equivalent to :WGEN1

### Reporting the Setup

Use :WGEN<w>? to query setup information for the WGEN<w> subsystem.

### Return Format

The following is a sample response from the :WGEN? query. In this case, the query was issued following the \*RST command.

```
:WGEN1:FUNC SIN;OUTP 0;FREQ +1.0000E+03;VOLT +500.0E-03;VOLT:OFFS
+0.0E+00;:WGEN1:OUTP:LOAD ONEM
```

**:WGEN<w>:ARBitrary:BYTeorder**

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:ARBitrary:BYTeorder <order>

<w> ::= 1 or 2 in NR1 format

<order> ::= {MSBFIRST | LSBFIRST}

The :WGEN<w>:ARBitrary:BYTeorder command selects the byte order for binary transfers.

**Query Syntax** :WGEN<w>:ARBitrary:BYTeorder?

The :WGEN<w>:ARBitrary:BYTeorder query returns the current byte order selection.

**Return Format** <order><NL>

<order> ::= {MSBFIRST | LSBFIRST}

**See Also** • "[:WGEN<w>:ARBitrary:DATA](#)" on page 1186  
• "[:WGEN<w>:ARBitrary:DATA:DAC](#)" on page 1189

## :WGEN<w>:ARBitrary:DATA

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:ARBitrary:DATA {<binary> | <value>, <value> ...}`

`<w> ::= 1 or 2 in NR1 format`

`<binary> ::= floating point values between -1.0 to +1.0  
                  in IEEE 488.2 binary block format`

`<value> ::= floating point values between -1.0 to +1.0  
                  in comma-separated format`

The :WGEN<w>:ARBitrary:DATA command downloads an arbitrary waveform in floating-point values format.

**See Also**

- [":WGEN<w>:ARBitrary:DATA:DAC"](#) on page 1189

- [":SAVE:ARBitrary\[:STARt\]"](#) on page 697

- [":RECall:ARBitrary\[:STARt\]"](#) on page 685

**:WGEN<w>:ARBitrary:DATA:ATTRibute:POINts**

**N** (see [page 1334](#))

**Query Syntax** :WGEN<w>:ARBitrary:DATA:ATTRibute:POINts?

<w> ::= 1 or 2 in NR1 format

The :WGEN<w>:ARBitrary:DATA:ATTRibute:POINts query returns the number of points used by the current arbitrary waveform.

**Return Format** <points> ::= number of points in NR1 format

- See Also**
- "[:WGEN<w>:ARBitrary:DATA](#)" on page 1186
  - "[:WGEN<w>:ARBitrary:DATA:DAC](#)" on page 1189
  - "[:SAVE:ARBitrary\[:STARt\]](#)" on page 697
  - "[:RECall:ARBitrary\[:STARt\]](#)" on page 685

**:WGEN<w>:ARBitrary:DATA:CLEar****N** (see [page 1334](#))**Command Syntax**    `:WGEN<w>:ARBitrary:DATA:CLEar``<w> ::= 1 or 2 in NR1 format`

The `:WGEN<w>:ARBitrary:DATA:CLEar` command clears the arbitrary waveform memory and loads it with the default waveform.

**See Also**

- "[":WGEN<w>:ARBitrary:DATA](#)" on page 1186
- "[":WGEN<w>:ARBitrary:DATA:DAC](#)" on page 1189
- "[":SAVE:ARBitrary\[:STARt\]](#)" on page 697
- "[":RECall:ARBitrary\[:STARt\]](#)" on page 685

## :WGEN<w>:ARBitrary:DATA:DAC

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:ARBitrary:DATA:DAC {<binary> | <value>, <value> ...}

<w> ::= 1 or 2 in NR1 format

<binary> ::= decimal 16-bit integer values between -512 to +511  
in IEEE 488.2 binary block format

<value> ::= decimal integer values between -512 to +511  
in comma-separated NR1 format

The :WGEN<w>:ARBitrary:DATA:DAC command downloads an arbitrary waveform using 16-bit integer (DAC) values.

**See Also**

- "[:WGEN<w>:ARBitrary:DATA](#)" on page 1186
- "[:SAVE:ARBitrary\[:STARt\]](#)" on page 697
- "[:RECall:ARBitrary\[:STARt\]](#)" on page 685

## :WGEN<w>:ARBitrary:INTerpolate

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:ARBitrary:INTerpolate {{0 | OFF} | {1 | ON}}`  
`<w> ::= 1 or 2 in NR1 format`

The :WGEN<w>:ARBitrary:INTerpolate command enables or disables the Interpolation control.

Interpolation specifies how lines are drawn between arbitrary waveform points:

- When ON, lines are drawn between points in the arbitrary waveform. Voltage levels change linearly between one point and the next.
- When OFF, all line segments in the arbitrary waveform are horizontal. The voltage level of one point remains until the next point.

**Query Syntax**    `:WGEN<w>:ARBitrary:INTerpolate?`

The :WGEN<w>:ARBitrary:INTerpolate query returns the current interpolation setting.

**Return Format**    `{0 | 1}`

**See Also**

- "[:WGEN<w>:ARBitrary:DATA](#)" on page 1186
- "[:WGEN<w>:ARBitrary:DATA:DAC](#)" on page 1189
- "[:SAVE:ARBitrary\[:START\]](#)" on page 697
- "[:RECall:ARBitrary\[:START\]](#)" on page 685

## :WGEN<w>:ARBitrary:STORe

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:ARBitrary:STORe <source>

```

<w> ::= 1 or 2 in NR1 format
<source> ::= {CHANnel<n> | WMEMory<r> | FUNCtion<m> | MATH<m> | FFT}
<n> ::= 1 to (# analog channels) in NR1 format
<r> ::= 1 to (# ref waveforms) in NR1 format
<m> ::= 1 to (# math functions) in NR1 format

```

The :WGEN<w>:ARBitrary:STORe command stores the source's waveform into the arbitrary waveform memory.

**See Also**

- "[:WGEN<w>:ARBitrary:DATA](#)" on page 1186
- "[:WGEN<w>:ARBitrary:DATA:DAC](#)" on page 1189
- "[:SAVE:ARBitrary\[:STARt\]](#)" on page 697
- "[:RECall:ARBitrary\[:STARt\]](#)" on page 685

**:WGEN<w>:FREQuency****N** (see [page 1334](#))**Command Syntax**    `:WGEN<w>:FREQuency <frequency>`    `<w> ::= 1 or 2 in NR1 format`    `<frequency> ::= frequency in Hz in NR3 format`

For all waveforms except Noise and DC, the :WGEN<w>:FREQuency command specifies the frequency of the waveform.

You can also specify the frequency indirectly using the :WGEN<w>:PERiod command.

**Query Syntax**    `:WGEN<w>:FREQuency?`

The :WGEN<w>:FREQuency? query returns the currently set waveform generator frequency.

**Return Format**    `<frequency><NL>`    `<frequency> ::= frequency in Hz in NR3 format`**See Also**

- "Introduction to :WGEN<w> Commands" on page 1184
- ":WGEN<w>:FUNCTION" on page 1193
- ":WGEN<w>:PERiod" on page 1217

## :WGEN<w>:FUNCTION

**N** (see page 1334)

**Command Syntax** :WGEN<w>:FUNCTION <signal>

<w> ::= 1 or 2 in NR1 format

<signal> ::= {SINusoid | SQUare | RAMP | PULSe | DC | NOISE | SINC  
| EXPRIse | EXPFall | CARDiac | GAUSSian | ARBITrary}

The :WGEN<w>:FUNCTION command selects the type of waveform:

Waveform Type	Characteristics	Frequency Range	Max. Amplitude <sup>2</sup> (High-Z) <sup>1</sup>	Offset <sup>2</sup> (High-Z) <sup>1</sup>
SINusoid	<p>Use these commands to set the sine signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> </ul>	100 mHz to 20 MHz	20 mVpp to 10 Vpp	±4.00 V
SQUare	<p>Use these commands to set the square wave signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:FUNCTION:SQUare:DCYCLE"</a> on page 1199</li> </ul> <p>The duty cycle can be adjusted from 20% to 80%.</p>	100 mHz to 10 MHz	20 mVpp to 10 Vpp	±5.00 V

Waveform Type	Characteristics	Frequency Range	Max. Amplitude <sup>2</sup> (High-Z) <sup>1</sup>	Offset <sup>2</sup> (High-Z) <sup>1</sup>
RAMP	<p>Use these commands to set the ramp signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:FUNCTION:RAMP:SYMMetry"</a> on page 1198</li> </ul> <p>Symmetry represents the amount of time per cycle that the ramp waveform is rising and can be adjusted from 0% to 100%.</p>	100 mHz to 200 kHz	20 mVpp to 10 Vpp	±5.00 V
PULSe	<p>Use these commands to set the pulse signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:FUNCTION:PULSe:WIDTH"</a> on page 1197</li> </ul> <p>The pulse width can be adjusted from 20 ns to the period minus 20 ns.</p>	100 mHz to 10 MHz	20 mVpp to 10 Vpp	±5.00 V
DC	<p>Use this command to set the DC level:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> </ul>	n/a	n/a	±10.00 V
NOiSe	<p>Use these commands to set the noise signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> </ul>	n/a	20 mVpp to 10 Vpp	±5.00 V

Waveform Type	Characteristics	Frequency Range	Max. Amplitude <sup>2</sup> (High-Z) <sup>1</sup>	Offset <sup>2</sup> (High-Z) <sup>1</sup>
SINC	<p>Use these commands to set the sine cardinal signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> </ul>	100 mHz to 1 MHz	20 mVpp to 9 Vpp	±2.50 V
EXPRIse	<p>Use these commands to set the exponential rise signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> </ul>	100 mHz to 5 MHz	20 mVpp to 10 Vpp	±5.00 V
EXPFall	<p>Use these commands to set the exponential fall signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:HIGH"</a> on page 1220</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:LOW"</a> on page 1221</li> </ul>	100 mHz to 5 MHz	20 mVpp to 10 Vpp	±5.00 V
CARDiac	<p>Use these commands to set the cardiac signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> </ul>	100 mHz to 200 kHz	20 mVpp to 9 Vpp	±2.50 V
GAUSSian	<p>Use these commands to set the gaussian pulse signal parameters:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">":WGEN&lt;w&gt;:FREQuency"</a> on page 1192</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:PERiod"</a> on page 1217</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage"</a> on page 1219</li> <li>▪ <a href="#">":WGEN&lt;w&gt;:VOLTage:OFFSet"</a> on page 1222</li> </ul>	100 mHz to 5 MHz	20 mVpp to 7.5 Vpp	±2.50 V

Waveform Type	Characteristics	Frequency Range	Max. Amplitude <sup>2</sup> (High-Z) <sup>1</sup>	Offset <sup>2</sup> (High-Z) <sup>1</sup>
ARbitrary	<p>Use these commands to set the arbitrary signal parameters:</p> <ul style="list-style-type: none"> <li>▪ "<a href="#">:WGEN&lt;w&gt;:FREQuency</a>" on page 1192</li> <li>▪ "<a href="#">:WGEN&lt;w&gt;:PERiod</a>" on page 1217</li> <li>▪ "<a href="#">:WGEN&lt;w&gt;:VOLTage</a>" on page 1219</li> <li>▪ "<a href="#">:WGEN&lt;w&gt;:VOLTage:OFFSet</a>" on page 1222</li> <li>▪ "<a href="#">:WGEN&lt;w&gt;:VOLTage:HIGH</a>" on page 1220</li> <li>▪ "<a href="#">:WGEN&lt;w&gt;:VOLTage:LOW</a>" on page 1221</li> </ul>	100 mHz to 12 MHz	20 mVpp to 10 Vpp	±5.00 V

<sup>1</sup>When the output load is 50 Ω, these values are halved.

<sup>2</sup>The minimum amplitude is limited to 40 mVpp if the offset is greater than 500 mV or less than -500 mV. Likewise, the offset is limited to +/-500 mV if the amplitude is less than 40 mVpp.

**Query Syntax**    `:WGEN<w>:FUNCTION?`

The `:WGEN<w>:FUNCTION?` query returns the currently selected signal type.

**Return Format**    `<signal><NL>`

```
<signal> ::= {SIN | SQU | RAMP | PULS | DC | NOIS | SINC | EXPR | EXPF
               | CARD | GAUS | ARB}
```

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[:WGEN<w>:MODulation:NOISE](#)" on page 1208

## :WGEN<w>:FUNCTION:PULSe:WIDTH

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:FUNCTION:PULSe:WIDTH <width>

<w> ::= 1 or 2 in NR1 format

<width> ::= pulse width in seconds in NR3 format

For Pulse waveforms, the :WGEN<w>:FUNCTION:PULSe:WIDTH command specifies the width of the pulse.

The pulse width can be adjusted from 20 ns to the period minus 20 ns.

**Query Syntax** :WGEN<w>:FUNCTION:PULSe:WIDTH?

The :WGEN<w>:FUNCTION:PULSe:WIDTH? query returns the currently set pulse width.

**Return Format** <width><NL>

<width> ::= pulse width in seconds in NR3 format

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193

**:WGEN<w>:FUNCTION:RAMP:SYMMetry****N** (see [page 1334](#))**Command Syntax**    `:WGEN<w>:FUNCTION:RAMP:SYMMetry <percent>`    `<w> ::= 1 or 2 in NR1 format`    `<percent> ::= symmetry percentage from 0% to 100% in NR3 format`

For Ramp waveforms, the :WGEN<w>:FUNCTION:RAMP:SYMMetry command specifies the symmetry of the waveform.

Symmetry represents the amount of time per cycle that the ramp waveform is rising.

**Query Syntax**    `:WGEN<w>:FUNCTION:RAMP:SYMMetry?`

The :WGEN<w>:FUNCTION:RAMP:SYMMetry? query returns the currently set ramp symmetry.

**Return Format**    `<percent><NL>`    `<percent> ::= symmetry percentage from 0% to 100% in NR3 format`**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184

- "[":WGEN<w>:FUNCTION](#)" on page 1193

## :WGEN<w>:FUNCTION:SQUare:DCYCle

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:FUNCTION:SQUare:DCYCle <percent>`

`<w> ::= 1 or 2 in NR1 format`

`<percent> ::= duty cycle percentage from 20% to 80% in NR3 format`

For Square waveforms, the :WGEN<w>:FUNCTION:SQUare:DCYCle command specifies the square wave duty cycle.

Duty cycle is the percentage of the period that the waveform is high.

**Query Syntax**    `:WGEN<w>:FUNCTION:SQUare:DCYCle?`

The :WGEN<w>:FUNCTION:SQUare:DCYCle? query returns the currently set square wave duty cycle.

**Return Format**    `<percent><NL>`

`<percent> ::= duty cycle percentage from 20% to 80% in NR3 format`

**See Also**

- "Introduction to :WGEN<w> Commands" on page 1184
- "[:WGEN<w>:FUNCTION](#)" on page 1193

## :WGEN<w>:MODulation:AM:DEPTH

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:AM:DEPTH &lt;percent&gt;</code>
	<code>&lt;w&gt; ::= 1 in NR1 format</code>
	<code>&lt;percent&gt; ::= AM depth percentage from 0% to 100% in NR1 format</code>

The :WGEN<w>:MODulation:AM:DEPTH command specifies the amount of amplitude modulation.

AM Depth refers to the portion of the amplitude range that will be used by the modulation. For example, a depth setting of 80% causes the output amplitude to vary from 10% to 90% (90% – 10% = 80%) of the original amplitude as the modulating signal goes from its minimum to maximum amplitude.

<b>Query Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:AM:DEPTH?</code>
---------------------	--------------------------------------------------

The :WGEN<w>:MODulation:AM:DEPTH? query returns the AM depth percentage setting.

<b>Return Format</b>	<code>&lt;percent&gt;&lt;NL&gt;</code>
	<code>&lt;percent&gt; ::= AM depth percentage from 0% to 100% in NR1 format</code>

<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:AM:FREQuency</a>" on page 1201</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FM:DEViation</a>" on page 1202</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FM:FREQuency</a>" on page 1203</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FSKey:FREQuency</a>" on page 1204</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FSKey:RATE</a>" on page 1205</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FUNCTION</a>" on page 1206</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FUNCTION:RAMP:SYMMetry</a>" on page 1207</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:STATE</a>" on page 1209</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:TYPE</a>" on page 1210</li> </ul>
-----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## :WGEN<w>:MODulation:AM:FREQuency

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:AM:FREQuency &lt;frequency&gt;</code>
	<code>&lt;w&gt; ::= 1 in NR1 format</code>
	<code>&lt;frequency&gt; ::= modulating waveform frequency in Hz in NR3 format</code>
	The :WGEN<w>:MODulation:AM:FREQuency command specifies the frequency of the modulating signal.
<b>Query Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:AM:FREQuency?</code>
	The :WGEN<w>:MODulation:AM:FREQuency? query returns the frequency of the modulating signal.
<b>Return Format</b>	<code>&lt;frequency&gt;&lt;NL&gt;</code>
	<code>&lt;frequency&gt; ::= modulating waveform frequency in Hz in NR3 format</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:AM:DEPTH" on page 1200</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FM:DEViation" on page 1202</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FM:FREQuency" on page 1203</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FSKey:FREQuency" on page 1204</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FSKey:RATE" on page 1205</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FUNCTION" on page 1206</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:FUNCTION:RAMP:SYMMetry" on page 1207</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:STATE" on page 1209</a></li> <li>· <a href="#">":WGEN&lt;w&gt;:MODulation:TYPE" on page 1210</a></li> </ul>

## :WGEN<w>:MODulation:FM:DEViation

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:MODulation:FM:DEViation <frequency>`

`<w> ::= 1 in NR1 format`

`<frequency> ::= frequency deviation in Hz in NR3 format`

The :WGEN<w>:MODulation:FM:DEViation command specifies the frequency deviation from the original carrier signal frequency.

When the modulating signal is at its maximum amplitude, the output frequency is the carrier signal frequency plus the deviation amount, and when the modulating signal is at its minimum amplitude, the output frequency is the carrier signal frequency minus the deviation amount.

The frequency deviation cannot be greater than the original carrier signal frequency.

Also, the sum of the original carrier signal frequency and the frequency deviation must be less than or equal to the maximum frequency for the selected waveform generator function plus 100 kHz.

**Query Syntax**    `:WGEN<w>:MODulation:FM:DEViation?`

The :WGEN<w>:MODulation:FM:DEViation? query returns the frequency deviation setting.

**Return Format**    `<frequency><NL>`

`<frequency> ::= frequency deviation in Hz in NR3 format`

**See Also**

- [":WGEN<w>:MODulation:AM:DEPTH" on page 1200](#)
- [":WGEN<w>:MODulation:AM:FREQuency" on page 1201](#)
- [":WGEN<w>:MODulation:FM:FREQuency" on page 1203](#)
- [":WGEN<w>:MODulation:FSKey:FREQuency" on page 1204](#)
- [":WGEN<w>:MODulation:FSKey:RATE" on page 1205](#)
- [":WGEN<w>:MODulation:FUNCTION" on page 1206](#)
- [":WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry" on page 1207](#)
- [":WGEN<w>:MODulation:STATE" on page 1209](#)
- [":WGEN<w>:MODulation:TYPE" on page 1210](#)

## :WGEN<w>:MODulation:FM:FREQuency

**N** (see [page 1334](#))

<b>Command Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:FM:FREQuency &lt;frequency&gt;</code> <code>&lt;w&gt; ::= 1 in NR1 format</code> <code>&lt;frequency&gt; ::= modulating waveform frequency in Hz in NR3 format</code>
	The :WGEN<w>:MODulation:FM:FREQuency command specifies the frequency of the modulating signal.
<b>Query Syntax</b>	<code>:WGEN&lt;w&gt;:MODulation:FM:FREQuency?</code>
	The :WGEN<w>:MODulation:FM:FREQuency? query returns the frequency of the modulating signal.
<b>Return Format</b>	<code>&lt;frequency&gt;&lt;NL&gt;</code> <code>&lt;frequency&gt; ::= modulating waveform frequency in Hz in NR3 format</code>
<b>See Also</b>	<ul style="list-style-type: none"> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:AM:DEPTH</a>" on page 1200</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:AM:FREQuency</a>" on page 1201</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FM:DEViation</a>" on page 1202</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FSKey:FREQuency</a>" on page 1204</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FSKey:RATE</a>" on page 1205</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FUNCTION</a>" on page 1206</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:FUNCTION:RAMP:SYMMetry</a>" on page 1207</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:STATE</a>" on page 1209</li> <li>· "<a href="#">:WGEN&lt;w&gt;:MODulation:TYPE</a>" on page 1210</li> </ul>

## :WGEN<w>:MODulation:FSKey:FREQuency

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:MODulation:FSKey:FREQuency <frequency>`

`<w> ::= 1 in NR1 format`

`<frequency> ::= hop frequency in Hz in NR3 format`

The :WGEN<w>:MODulation:FSKey:FREQuency command specifies the "hop frequency".

The output frequency "shifts" between the original carrier frequency and this "hop frequency".

**Query Syntax**    `:WGEN<w>:MODulation:FSKey:FREQuency?`

The :WGEN<w>:MODulation:FSKey:FREQuency? query returns the "hop frequency" setting.

**Return Format**    `<frequency><NL>`

`<frequency> ::= hop frequency in Hz in NR3 format`

**See Also**

- [":WGEN<w>:MODulation:AM:DEPTH" on page 1200](#)
- [":WGEN<w>:MODulation:AM:FREQuency" on page 1201](#)
- [":WGEN<w>:MODulation:FM:DEViation" on page 1202](#)
- [":WGEN<w>:MODulation:FM:FREQuency" on page 1203](#)
- [":WGEN<w>:MODulation:FSKey:RATE" on page 1205](#)
- [":WGEN<w>:MODulation:FUNCTION" on page 1206](#)
- [":WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry" on page 1207](#)
- [":WGEN<w>:MODulation:STATE" on page 1209](#)
- [":WGEN<w>:MODulation:TYPE" on page 1210](#)

## :WGEN<w>:MODulation:FSKey:RATE

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:MODulation:FSKey:RATE <rate>

<w> ::= 1 in NR1 format

<rate> ::= FSK modulation rate in Hz in NR3 format

The :WGEN<w>:MODulation:FSKey:RATE command specifies the rate at which the output frequency "shifts".

The FSK rate specifies a digital square wave modulating signal.

**Query Syntax** :WGEN<w>:MODulation:FSKey:RATE?

The :WGEN<w>:MODulation:FSKey:RATE? query returns the FSK rate setting.

**Return Format** <rate><NL>

<rate> ::= FSK modulation rate in Hz in NR3 format

- See Also**
- "[:WGEN<w>:MODulation:AM:DEPTH](#)" on page 1200
  - "[:WGEN<w>:MODulation:AM:FREQuency](#)" on page 1201
  - "[:WGEN<w>:MODulation:FM:DEViation](#)" on page 1202
  - "[:WGEN<w>:MODulation:FM:FREQuency](#)" on page 1203
  - "[:WGEN<w>:MODulation:FSKey:FREQuency](#)" on page 1204
  - "[:WGEN<w>:MODulation:FUNCTION](#)" on page 1206
  - "[:WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry](#)" on page 1207
  - "[:WGEN<w>:MODulation:STATE](#)" on page 1209
  - "[:WGEN<w>:MODulation:TYPE](#)" on page 1210

## :WGEN<w>:MODulation:FUNCTION

**N** (see [page 1334](#))

**Command Syntax**

```
:WGEN<w>:MODulation:FUNCTION <shape>
<w> ::= 1 in NR1 format
<shape> ::= {SINusoid | SQUare| RAMP}
```

The :WGEN<w>:MODulation:FUNCTION command specifies the shape of the modulating signal.

When the RAMP shape is selected, you can specify the amount of time per cycle that the ramp waveform is rising with the :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry command.

This command applies to AM and FM modulation. (The FSK modulation signal is a square wave shape.)

**Query Syntax**

```
:WGEN<w>:MODulation:FUNCTION?
```

The :WGEN<w>:MODulation:FUNCTION? query returns the specified modulating signal shape.

**Return Format**

```
<shape><NL>
<shape> ::= {SIN | SQU| RAMP}
```

**See Also**

- "[":WGEN<w>:MODulation:AM:DEPTH](#)" on page 1200
- "[":WGEN<w>:MODulation:AM:FREQuency](#)" on page 1201
- "[":WGEN<w>:MODulation:FM:DEViation](#)" on page 1202
- "[":WGEN<w>:MODulation:FM:FREQuency](#)" on page 1203
- "[":WGEN<w>:MODulation:FSKey:FREQuency](#)" on page 1204
- "[":WGEN<w>:MODulation:FSKey:RATE](#)" on page 1205
- "[":WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry](#)" on page 1207
- "[":WGEN<w>:MODulation:STATe](#)" on page 1209
- "[":WGEN<w>:MODulation:TYPE](#)" on page 1210

## :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry <percent>

<w> ::= 1 in NR1 format

<percent> ::= symmetry percentage from 0% to 100% in NR1 format

The :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry command specifies the amount of time per cycle that the ramp waveform is rising. The ramp modulating waveform shape is specified with the :WGEN<w>:MODulation:FUNCTION command.

**Query Syntax** :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry?

The :WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry? query returns ramp symmetry percentage setting.

**Return Format** <percent><NL>

<percent> ::= symmetry percentage from 0% to 100% in NR1 format

**See Also**

- "[:WGEN<w>:MODulation:AM:DEPTH](#)" on page 1200
- "[:WGEN<w>:MODulation:AM:FREQuency](#)" on page 1201
- "[:WGEN<w>:MODulation:FM:DEViation](#)" on page 1202
- "[:WGEN<w>:MODulation:FM:FREQuency](#)" on page 1203
- "[:WGEN<w>:MODulation:FSKey:FREQuency](#)" on page 1204
- "[:WGEN<w>:MODulation:FSKey:RATE](#)" on page 1205
- "[:WGEN<w>:MODulation:FUNCTION](#)" on page 1206
- "[:WGEN<w>:MODulation:STATe](#)" on page 1209
- "[:WGEN<w>:MODulation:TYPE](#)" on page 1210

**:WGEN<w>:MODulation:NOISe****N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:MODulation:NOISe <percent>`  
`<w> ::= 1 or 2 in NR1 format`  
`<percent> ::= 0 to 100`

The :WGEN<w>:MODulation:NOISe command adds noise to the currently selected signal. The sum of the amplitude between the original signal and injected noise is limited to the regular amplitude limit (for example, 5 Vpp in 1 MOhm), so the range for <percent> varies according to current amplitude.

Note that adding noise affects edge triggering on the waveform generator source as well as the waveform generator sync pulse output signal (which can be sent to TRIG OUT). This is because the trigger comparator is located after the noise source.

**Query Syntax**    `:WGEN<w>:MODulation:NOISe?`

The :WGEN<w>:MODulation:NOISe query returns the percent of added noise.

**Return Format**    `<percent><NL>`  
`<percent> ::= 0 to 100`

**See Also**    • [":WGEN<w>:FUNCTION"](#) on page 1193

## :WGEN<w>:MODulation:STATe

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:MODulation:STATe <setting>

<w> ::= 1 in NR1 format

<setting> ::= {{OFF | 0} | {ON | 1}}

The :WGEN<w>:MODulation:STATe command enables or disables modulated waveform generator output.

You can enable modulation for all waveform generator function types except pulse, DC, and noise.

**Query Syntax** :WGEN<w>:MODulation:STATe?

The :WGEN<w>:MODulation:STATe? query returns whether the modulated waveform generator output is enabled or disabled.

**Return Format** <setting><NL>

<setting> ::= {0 | 1}

**See Also**

- "[:WGEN<w>:MODulation:AM:DEPTH](#)" on page 1200
- "[:WGEN<w>:MODulation:AM:FREQuency](#)" on page 1201
- "[:WGEN<w>:MODulation:FM:DEViation](#)" on page 1202
- "[:WGEN<w>:MODulation:FM:FREQuency](#)" on page 1203
- "[:WGEN<w>:MODulation:FSKey:FREQuency](#)" on page 1204
- "[:WGEN<w>:MODulation:FSKey:RATE](#)" on page 1205
- "[:WGEN<w>:MODulation:FUNCTION](#)" on page 1206
- "[:WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry](#)" on page 1207
- "[:WGEN<w>:MODulation:TYPE](#)" on page 1210

## :WGEN<w>:MODulation:TYPE

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:MODulation:TYPE <type>`

`<w> ::= 1 in NR1 format`

`<type> ::= {AM | FM | FSK}`

The :WGEN<w>:MODulation:TYPE command selects the modulation type:

- AM (amplitude modulation) – the amplitude of the original carrier signal is modified according to the amplitude of the modulating signal.

Use the :WGEN<w>:MODulation:AM:FREQuency command to set the modulating signal frequency.

Use the :WGEN<w>:MODulation:AM:DEPTh command to specify the amount of amplitude modulation.

- FM (frequency modulation) – the frequency of the original carrier signal is modified according to the amplitude of the modulating signal.

Use the :WGEN<w>:MODulation:FM:FREQuency command to set the modulating signal frequency.

Use the :WGEN<w>:MODulation:FM:DEViation command to specify the frequency deviation from the original carrier signal frequency.

- FSK (frequency-shift keying modulation) – the output frequency "shifts" between the original carrier frequency and a "hop frequency" at the specified FSK rate.

The FSK rate specifies a digital square wave modulating signal.

Use the :WGEN<w>:MODulation:FSKey:FREQuency command to specify the "hop frequency".

Use the :WGEN<w>:MODulation:FSKey:RATE command to specify the rate at which the output frequency "shifts".

**Query Syntax**    `:WGEN<w>:MODulation:TYPE?`

The :WGEN<w>:MODulation:TYPE? query returns the selected modulation type.

**Return Format**    `<type><NL>`

`<type> ::= {AM | FM | FSK}`

**See Also**    [":WGEN<w>:MODulation:AM:DEPTh" on page 1200](#)

[":WGEN<w>:MODulation:AM:FREQuency" on page 1201](#)

[":WGEN<w>:MODulation:FM:DEViation" on page 1202](#)

[":WGEN<w>:MODulation:FM:FREQuency" on page 1203](#)

[":WGEN<w>:MODulation:FSKey:FREQuency" on page 1204](#)

- [":WGEN<w>:MODulation:FSKey:RATE"](#) on page 1205
- [":WGEN<w>:MODulation:FUNCTION"](#) on page 1206
- [":WGEN<w>:MODulation:FUNCTION:RAMP:SYMMetry"](#) on page 1207
- [":WGEN<w>:MODulation:STATE"](#) on page 1209

**:WGEN<w>:OUTPut****N** (see [page 1334](#))**Command Syntax**    `:WGEN<w>:OUTPut <on_off>`    `<w> ::= 1 or 2 in NR1 format`    `<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :WGEN<w>:OUTPut command specifies whether the waveform generator signal output is ON (1) or OFF (0).

**Query Syntax**    `:WGEN<w>:OUTPut?`

The :WGEN<w>:OUTPut? query returns the current state of the waveform generator output setting.

**Return Format**    `<on_off><NL>`    `<on_off> ::= {1 | 0}`**See Also**    • ["Introduction to :WGEN<w> Commands"](#) on page 1184

## :WGEN<w>:OUTPut:LOAD

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:OUTPut:LOAD <impedance>`  
`<w> ::= 1 or 2 in NR1 format`  
`<impedance> ::= {ONEMeg | FIFTy}`

The :WGEN<w>:OUTPut:LOAD command selects the expected output load impedance.

The output impedance of the Gen Out BNC is fixed at 50 ohms. However, the output load selection lets the waveform generator display the correct amplitude and offset levels for the expected output load.

If the actual load impedance is different than the selected value, the displayed amplitude and offset levels will be incorrect.

**Query Syntax**    `:WGEN<w>:OUTPut:LOAD?`

The :WGEN<w>:OUTPut:LOAD? query returns the current expected output load impedance.

**Return Format**    `<impedance><NL>`  
`<impedance> ::= {ONEM | FIFT}`

**See Also**    · ["Introduction to :WGEN<w> Commands"](#) on page 1184

**:WGEN<w>:OUTPut:MODE**(see [page 1334](#))

**Command Syntax**    `:WGEN<w>:OUTPut:MODE <mode>`  
                  `<mode> ::= {NORMAL | SINGLE}`

The :WGEN<w>:OUTPut:MODE command specifies whether the defined waveform is output continuously or as a single cycle (single-shot):

- NORMAL – the defined waveform is output continuously.
- SINGLE – one cycle of the defined waveform is output when you send the :WGEN<w>:OUTPut:SINGLe command.  
  
Not all waveform types allow single-shot output.

**Query Syntax**    `:WGEN<w>:OUTPut:MODE?`

The :WGEN<w>:OUTPut:MODE? query returns the output mode setting.

**Return Format**    `<mode><NL>`  
                  `<mode> ::= {NORMAL | SINGLE}`

**See Also**    • "":WGEN<w>:OUTPut:SINGLe" on page 1216

## :WGEN<w>:OUTPut:POLarity

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:OUTPut:POLarity <polarity>

<w> ::= 1 or 2 in NR1 format

<polarity> ::= {NORMAL | INVERTed}

The :WGEN<w>:OUTPut:POLarity command specifies whether the waveform generator output is inverted..

**Query Syntax** :WGEN<w>:OUTPut:POLarity?

The :WGEN<w>:OUTPut:POLarity? query returns the specified output polarity.

**Return Format** <polarity><NL>

<polarity> ::= {NORM | INV}

**See Also** • ["Introduction to :WGEN<w> Commands"](#) on page 1184

## :WGEN<w>:OUTPut:SINGle



(see [page 1334](#))

**Command Syntax**    `:WGEN<w>:OUTPut:SINGle`

When the single-shot output mode is selected (by the :WGEN<w>:OUTPut:MODE command), the :WGEN<w>:OUTPut:SINGle command causes a single cycle of the defined waveform to be output.

Sending this command multiple times will interrupt a slow signal output before the cycle is completed.

**See Also**    - [":WGEN<w>:OUTPut:MODE"](#) on page 1214

## :WGEN<w>:PERiod

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:PERiod <period>

<w> ::= 1 or 2 in NR1 format

<period> ::= period in seconds in NR3 format

For all waveforms except Noise and DC, the :WGEN<w>:PERiod command specifies the period of the waveform.

You can also specify the period indirectly using the :WGEN<w>:FREQuency command.

**Query Syntax** :WGEN<w>:PERiod?

The :WGEN<w>:PERiod? query returns the currently set waveform generator period.

**Return Format** <period><NL>

<period> ::= period in seconds in NR3 format

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[:WGEN<w>:FUNCTION](#)" on page 1193
- "[:WGEN<w>:FREQuency](#)" on page 1192

## :WGEN<w>:RST

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:RST`

`<w> ::= 1 or 2 in NR1 format`

The `:WGEN<w>:RST` command restores the waveform generator factory default settings (1 kHz sine wave, 500 mVpp, 0 V offset).

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193
- "[":WGEN<w>:FREQuency](#)" on page 1192

## :WGEN<w>:VOLTage

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:VOLTage <amplitude>

<w> ::= 1 or 2 in NR1 format

<amplitude> ::= amplitude in volts in NR3 format

For all waveforms except DC, the :WGEN<w>:VOLTage command specifies the waveform's amplitude. Use the :WGEN<w>:VOLTage:OFFSet command to specify the offset voltage or DC level.

You can also specify the amplitude and offset indirectly using the :WGEN<w>:VOLTage:HIGH and :WGEN<w>:VOLTage:LOW commands. For example, an amplitude of 5 V and an offset of 1 V is the same as a high-level voltage of 4 V and a low-level voltage of -1 V.

### NOTE

When the amplitude is below 40 mV, the offset is limited to  $\pm 500$  mV.

### Query Syntax

:WGEN<w>:VOLTage?

The :WGEN<w>:VOLTage? query returns the currently specified waveform amplitude.

### Return Format

<amplitude><NL>

<amplitude> ::= amplitude in volts in NR3 format

### See Also

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193
- "[":WGEN<w>:VOLTage:OFFSet](#)" on page 1222
- "[":WGEN<w>:VOLTage:HIGH](#)" on page 1220
- "[":WGEN<w>:VOLTage:LOW](#)" on page 1221

## :WGEN<w>:VOLTage:HIGH

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:VOLTage:HIGH <high>`

`<w> ::= 1 or 2 in NR1 format`

`<high> ::= high-level voltage in volts, in NR3 format`

For all waveforms except DC, the :WGEN<w>:VOLTage:HIGH command specifies the waveform's high-level voltage. Use the :WGEN<w>:VOLTage:LOW command to specify the low-level voltage.

You can also specify the high-level and low-level voltages indirectly using the :WGEN<w>:VOLTage and :WGEN<w>:VOLTage:OFFSET commands. For example, a high-level voltage of 4 V and a low-level voltage of -1 V is the same as an amplitude of 5 V and an offset of 1 V.

### NOTE

When the amplitude is below 40 mV, the offset is limited to  $\pm 500$  mV.

### Query Syntax

`:WGEN<w>:VOLTage:HIGH?`

The :WGEN<w>:VOLTage:HIGH? query returns the currently specified waveform high-level voltage.

### Return Format

`<high><NL>`

`<high> ::= high-level voltage in volts, in NR3 format`

### See Also

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193
- "[":WGEN<w>:VOLTage:LOW](#)" on page 1221
- "[":WGEN<w>:VOLTage](#)" on page 1219
- "[":WGEN<w>:VOLTage:OFFSET](#)" on page 1222

## :WGEN<w>:VOLTage:LOW

**N** (see [page 1334](#))

**Command Syntax** :WGEN<w>:VOLTage:LOW <low>

<w> ::= 1 or 2 in NR1 format

<low> ::= low-level voltage in volts, in NR3 format

For all waveforms except DC, the :WGEN<w>:VOLTage:LOW command specifies the waveform's low-level voltage. Use the :WGEN<w>:VOLTage:HIGH command to specify the high-level voltage.

You can also specify the high-level and low-level voltages indirectly using the :WGEN<w>:VOLTage and :WGEN<w>:VOLTage:OFFSET commands. For example, a high-level voltage of 4 V and a low-level voltage of -1 V is the same as an amplitude of 5 V and an offset of 1 V.

### NOTE

When the amplitude is below 40 mV, the offset is limited to  $\pm 500$  mV.

**Query Syntax** :WGEN<w>:VOLTage:LOW?

The :WGEN<w>:VOLTage:LOW? query returns the currently specified waveform low-level voltage.

**Return Format** <low><NL>

<low> ::= low-level voltage in volts, in NR3 format

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193
- "[":WGEN<w>:VOLTage:LOW](#)" on page 1221
- "[":WGEN<w>:VOLTage](#)" on page 1219
- "[":WGEN<w>:VOLTage:OFFSET](#)" on page 1222

## :WGEN<w>:VOLTage:OFFSet

**N** (see [page 1334](#))

**Command Syntax**    `:WGEN<w>:VOLTage:OFFSet <offset>`

`<w> ::= 1 or 2 in NR1 format`

`<offset> ::= offset in volts in NR3 format`

The :WGEN<w>:VOLTage:OFFSet command specifies the waveform's offset voltage or the DC level. Use the :WGEN<w>:VOLTage command to specify the amplitude.

You can also specify the amplitude and offset indirectly using the :WGEN<w>:VOLTage:HIGH and :WGEN<w>:VOLTage:LOW commands. For example, an amplitude of 5 V and an offset of 1 V is the same as a high-level voltage of 4 V and a low-level voltage of -1 V.

### NOTE

When the amplitude is below 40 mV, the offset is limited to  $\pm 500$  mV.

**Query Syntax**    `:WGEN<w>:VOLTage:OFFSet?`

The :WGEN<w>:VOLTage:OFFSet? query returns the currently specified waveform offset voltage.

**Return Format**    `<offset><NL>`

`<offset> ::= offset in volts in NR3 format`

**See Also**

- "[Introduction to :WGEN<w> Commands](#)" on page 1184
- "[":WGEN<w>:FUNCTION](#)" on page 1193
- "[":WGEN<w>:VOLTage](#)" on page 1219
- "[":WGEN<w>:VOLTage:HIGH](#)" on page 1220
- "[":WGEN<w>:VOLTage:LOW](#)" on page 1221

# 36 :WMEMory<r> Commands

Control reference waveforms.

**Table 156:**:WMEMory<r> Commands Summary

Command	Query	Options and Query Returns
:WMEMory<r>:CLEar (see <a href="#">page 1225</a> )	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format
:WMEMory<r>:DISPlay { {0   OFF}   {1   ON} } (see <a href="#">page 1226</a> )	:WMEMory<r>:DISPLAY? (see <a href="#">page 1226</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format {0   1}
:WMEMory<r>:LABEL <string> (see <a href="#">page 1227</a> )	:WMEMory<r>:LABEL? (see <a href="#">page 1227</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <string> ::= any series of 10 or less ASCII characters enclosed in quotation marks
:WMEMory<r>:SAVE <source> (see <a href="#">page 1228</a> )	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format <source> ::= {CHANnel<n>   FUNCtion<m>   MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format NOTE: Only ADD or SUBtract math operations can be saved as reference waveforms.
:WMEMory<r>:SKEW <skew> (see <a href="#">page 1229</a> )	:WMEMory<r>:SKEW? (see <a href="#">page 1229</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <skew> ::= time in seconds in NR3 format
:WMEMory<r>:YOFFset <offset>[suffix] (see <a href="#">page 1230</a> )	:WMEMory<r>:YOFFset? (see <a href="#">page 1230</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <offset> ::= vertical offset value in NR3 format [suffix] ::= {V   mV}

**Table 156**:WMEMory<r> Commands Summary (continued)

Command	Query	Options and Query Returns
:WMEMory<r>:YRANGE <range>[suffix] (see <a href="#">page 1231</a> )	:WMEMory<r>:YRANGE? (see <a href="#">page 1231</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <range> ::= vertical full-scale range value in NR3 format [suffix] ::= {V   mV}
:WMEMory<r>:YScale <scale>[suffix] (see <a href="#">page 1232</a> )	:WMEMory<r>:YScale? (see <a href="#">page 1232</a> )	<r> ::= 1 to (# ref waveforms) in NR1 format <scale> ::= vertical units per division value in NR3 format [suffix] ::= {V   mV}

## :WMEMory<r>:CLEar

**N** (see [page 1334](#))

**Command Syntax** :WMEMory<r>:CLEar

```
<r> ::= 1 to (# ref waveforms) in NR1 format
```

The :WMEMory<r>:CLEar command clears the specified reference waveform location.

**See Also**

- [Chapter 36, “:WMEMory<r> Commands,” starting on page 1223](#)
- [":WMEMory<r>:SAVE" on page 1228](#)
- [":WMEMory<r>:DISPLAY" on page 1226](#)

## :WMEMory<r>:DISPlay

**N** (see [page 1334](#))

**Command Syntax**    `:WMEMory<r>:DISPlay <on_off>`

`<r> ::= 1 to (# ref waveforms) in NR1 format`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :WMEMory<r>:DISPlay command turns the display of the specified reference waveform on or off.

There are two reference waveform locations, but only one reference waveform can be displayed at a time. That means, if :WMEMory1:DISPlay is ON, sending the :WMEMory2:DISPlay ON command will automatically set :WMEMory1:DISPlay OFF.

**Query Syntax**    `:WMEMory<r>:DISPlay?`

The :WMEMory<r>:DISPlay? query returns the current display setting for the reference waveform.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- [Chapter 36](#), “:WMEMory<r> Commands,” starting on page 1223
- [":WMEMory<r>:CLEar"](#) on page 1225
- [":WMEMory<r>:LABEL"](#) on page 1227

## :WMEMory<r>:LABel

**N** (see [page 1334](#))

**Command Syntax** :WMEMory<r>:LABel <string>

<r> ::= 1 to (# ref waveforms) in NR1 format

<string> ::= quoted ASCII string

### NOTE

Label strings are 10 characters or less, and may contain any commonly used ASCII characters. Labels with more than 10 characters are truncated to 10 characters. Lower case characters are converted to upper case.

The :WMEMory<r>:LABel command sets the reference waveform label to the string that follows.

Setting a label for a reference waveform also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

**Query Syntax** :WMEMory<r>:LABel?

The :WMEMory<r>:LABel? query returns the label associated with a particular reference waveform.

**Return Format** <string><NL>

<string> ::= quoted ASCII string

**See Also**

- [Chapter 36](#), “:WMEMory<r> Commands,” starting on page 1223
- [":WMEMory<r>:DISPLAY"](#) on page 1226

## :WMEMory<r>:SAVE

**N** (see [page 1334](#))

**Command Syntax**    `:WMEMory<r>:SAVE <source>`

```
<r> ::= 1 to (# ref waveforms) in NR1 format  
<source> ::= {CHANnel<n> | FUNCTion<m> | MATH<m>}  
<n> ::= 1 to (# analog channels) in NR1 format  
<m> ::= 1 to (# math functions) in NR1 format
```

The :WMEMory<r>:SAVE command copies the analog channel or math function waveform to the specified reference waveform location.

### NOTE

Only ADD or SUBtract math operations can be saved as reference waveforms.

---

### See Also

- [Chapter 36](#), “:WMEMory<r> Commands,” starting on page 1223
- [":WMEMory<r>:DISPlay"](#) on page 1226

## :WMEMory<r>:SKEW

**N** (see [page 1334](#))

**Command Syntax** :WMEMory<r>:SKEW <skew>

```
<r> ::= 1 to (# ref waveforms) in NR1 format
<skew> ::= time in seconds in NR3 format
```

The :WMEMory<r>:SKEW command sets the skew factor for the specified reference waveform.

**Query Syntax** :WMEMory<r>:SKEW?

The :WMEMory<r>:SKEW? query returns the current skew setting for the selected reference waveform.

**Return Format** <skew><NL>

```
<skew> ::= time in seconds in NR3 format
```

**See Also**

- [Chapter 36](#), “:WMEMory<r> Commands,” starting on page 1223
- [":WMEMory<r>:DISPlay"](#) on page 1226
- [":WMEMory<r>:YOFFset"](#) on page 1230
- [":WMEMory<r>:YRANge"](#) on page 1231
- [":WMEMory<r>:YScale"](#) on page 1232

## :WMEMory<r>:YOFFset

**N** (see [page 1334](#))

**Command Syntax**    `:WMEMory<r>:YOFFset <offset> [<suffix>]`

`<r> ::= 1 to (# ref waveforms) in NR1 format`

`<offset> ::= vertical offset value in NR3 format`

`<suffix> ::= {v | mV}`

The :WMEMory<r>:YOFFset command sets the value that is represented at center screen for the selected reference waveform.

The range of legal values varies with the value set by the :WMEMory<r>:YRANge or :WMEMory<r>:YScale commands. If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value. Legal values are affected by the probe attenuation setting.

**Query Syntax**    `:WMEMory<r>:YOFFset?`

The :WMEMory<r>:YOFFset? query returns the current offset value for the selected reference waveform.

**Return Format**    `<offset><NL>`

`<offset> ::= vertical offset value in NR3 format`

**See Also**

- [Chapter 36, “:WMEMory<r> Commands,” starting on page 1223](#)
- [":WMEMory<r>:DISPlay" on page 1226](#)
- [":WMEMory<r>:YRANge" on page 1231](#)
- [":WMEMory<r>:YScale" on page 1232](#)
- [":WMEMory<r>:SKEW" on page 1229](#)

## :WMEMory<r>:YRANge

**N** (see [page 1334](#))

**Command Syntax** :WMEMory<r>:YRANge <range>[<suffix>]

<r> ::= 1 to (# ref waveforms) in NR1 format

<range> ::= vertical full-scale range value in NR3 format

<suffix> ::= {v | mV}

The :WMEMory<r>:YRANge command defines the full-scale vertical axis of the selected reference waveform.

Legal values for the range are copied from the original source waveform (that is, the analog channel or math function waveform that was originally saved as a reference waveform).

**Query Syntax** :WMEMory<r>:YRANge?

The :WMEMory<r>:YRANge? query returns the current full-scale range setting for the specified reference waveform.

**Return Format** <range><NL>

<range> ::= vertical full-scale range value in NR3 format

- See Also**
- [Chapter 36](#), “:WMEMory<r> Commands,” starting on page 1223
  - [":WMEMory<r>:DISPlay"](#) on page 1226
  - [":WMEMory<r>:YOFFset"](#) on page 1230
  - [":WMEMory<r>:SKEW"](#) on page 1229
  - [":WMEMory<r>:YScale"](#) on page 1232

## :WMEMory<r>:YSCale

**N** (see [page 1334](#))

**Command Syntax**    `:WMEMory<r>:YSCale <scale>[<suffix>]`

`<r>` ::= 1 to (# ref waveforms) in NR1 format

`<scale>` ::= vertical units per division in NR3 format

`<suffix>` ::= {v | mV}

The :WMEMory<r>:YSCale command sets the vertical scale, or units per division, of the selected reference waveform.

Legal values for the scale are copied from the original source waveform (that is, the analog channel or math function waveform that was originally saved as a reference waveform).

**Query Syntax**    `:WMEMory<r>:YSCale?`

The :WMEMory<r>:YSCale? query returns the current scale setting for the specified reference waveform.

**Return Format**    `<scale><NL>`

`<scale>` ::= vertical units per division in NR3 format

- See Also**
- [Chapter 36, “:WMEMory<r> Commands,” starting on page 1223](#)
  - [":WMEMory<r>:DISPlay" on page 1226](#)
  - [":WMEMory<r>:YOFFset" on page 1230](#)
  - [":WMEMory<r>:YRANge" on page 1231](#)
  - [":WMEMory<r>:SKEW" on page 1229](#)

# 37 Obsolete and Discontinued Commands

Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs (see "[Obsolete Commands](#)" on page 1334).

Obsolete Command	Current Command Equivalent	Behavior Differences
ANALog<n>:BWLimit	:CHANnel<n>:BWLimit (see <a href="#">page 282</a> )	
ANALog<n>:COUpling	:CHANnel<n>:COUpling (see <a href="#">page 283</a> )	
ANALog<n>:INVert	:CHANnel<n>:INVert (see <a href="#">page 286</a> )	
ANALog<n>:LABEL	:CHANnel<n>:LABEL (see <a href="#">page 287</a> )	
ANALog<n>:OFFSet	:CHANnel<n>:OFFSet (see <a href="#">page 288</a> )	
ANALog<n>:PROBe	:CHANnel<n>:PROBe (see <a href="#">page 289</a> )	
ANALog<n>:PMODe	none	
ANALog<n>:RANGE	:CHANnel<n>:RANGE (see <a href="#">page 296</a> )	
:CHANnel:ACTivity (see <a href="#">page 1240</a> )	:ACTivity (see <a href="#">page 207</a> )	
:CHANnel:LABEL (see <a href="#">page 1241</a> )	:CHANnel<n>:LABEL (see <a href="#">page 287</a> ) or :DIGital<n>:LABEL (see <a href="#">page 324</a> )	use CHANnel<n>:LABEL for analog channels and use DIGital<n>:LABEL for digital channels

Obsolete Command	Current Command Equivalent	Behavior Differences
:CHANnel:THReShold (see page 1242)	:POD<n>:THReShold (see page 605) or :DIGItal<d>:THReShold (see page 327)	
:CHANnel2:SKEW (see page 1243)	:CHANnel<n>:PROBe:SKEW (see page 293)	
:CHANnel<n>:INPut (see page 1244)	:CHANnel<n>:IMPedance (see page 285)	
:CHANnel<n>:PMODe (see page 1245)	none	
:DISPlay:CONNect (see page 1246)	:DISPlay:VECTors (see page 346)	
:DISPlay:ORDer (see page 1247)	none	
:ERASe (see page 1248)	:DISPlay:CLEar (see page 338)	
:EXternal:PMODe (see page 1249)	none	
FUNCTION1, FUNCTION2	:FUNCTION Commands (see page 379)	ADD not included
:FUNCTION Commands	:FUNCTION2 Commands (see page 379)	:FUNCTION commands (with no <m> number) map to :FUNCTION2. This allows legacy programs to work without change.
:FUNCTION:GOFT:OPERation (see page 1250)	:FUNCTION1:OPERation (see page 405)	GOFT maps to FUNCTION1.
:FUNCTION:GOFT:SOURce1 (see page 1251)	:FUNCTION1:SOURce1 (see page 413)	GOFT maps to FUNCTION1.
:FUNCTION:GOFT:SOURce2 (see page 1252)	:FUNCTION1:SOURce2 (see page 415)	GOFT maps to FUNCTION1.
:FUNCTION:SOURce (see page 1253)	:FUNCTION:SOURce1 (see page 413)	Obsolete command has ADD, SUBTract, and MULTiply parameters; current command has GOFT parameter.
:FUNCTION:VIEW (see page 1254)	:FUNCTION:DISPlay (see page 391)	
:HARDcopy:DESTination (see page 1255)	:HARDcopy:FILEname (see page 1256)	

Obsolete Command	Current Command Equivalent	Behavior Differences
:HARDcopy:FILEname (see page 1256)	:RECall:FILEname (see page 687) :SAVE:FILEname (see page 687)	
:HARDcopy:GRAYscale (see page 1257)	:HARDcopy:PALette (see page 431)	
:HARDcopy:IGColors (see page 1258)	:HARDcopy:INKSaver (see page 423)	
:HARDcopy:PDRiver (see page 1259)	:HARDcopy:APRinter (see page 420)	
:MEASure:LOWER (see page 1260)	:MEASure:DEFine:THResholds (see page 485)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:SCRatch (see page 1261)	:MEASure:CLEar (see page 482)	
:MEASure:TDELta (see page 1262)	:MARKer:XDELta (see page 450)	
:MEASure:THResholds (see page 1263)	:MEASure:DEFine:THResholds (see page 485)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:TMAX (see page 1264)	:MEASure:XMAX (see page 540)	
:MEASure:TMIN (see page 1265)	:MEASure:XMIN (see page 541)	
:MEASure:TSTArt (see page 1266)	:MARKer:X1Position (see page 445)	
:MEASure:TSTOP (see page 1267)	:MARKer:X2Position (see page 448)	
:MEASure:TVOLT (see page 1268)	:MEASure:TVALue (see page 527)	TVALue measures additional values such as db, Vs, etc.
:MEASure:UPPer (see page 1269)	:MEASure:DEFine:THResholds (see page 485)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:VDELta (see page 1270)	:MARKer:YDELta (see page 457)	
:MEASure:VSTArt (see page 1271)	:MARKer:Y1Position (see page 454)	
:MEASure:VSTOP (see page 1272)	:MARKer:Y2Position (see page 456)	

Obsolete Command	Current Command Equivalent	Behavior Differences
:MTEST:AMASK:{SAVE   STORe} (see page 1273)	:SAVE:MASK[:START] (see page 705)	
:MTEST:AVERage (see page 1274)	:ACQuire:TYPE AVERage (see page 255)	
:MTEST:AVERage:COUNt (see page 1275)	:ACQuire:COUNt (see page 246)	
:MTEST:LOAD (see page 1276)	:RECall:MASK[:STARt] (see page 689)	
:MTEST:RUMode (see page 1277)	:MTEST:RMODe (see page 586)	
:MTEST:RUMode:SOFailure (see page 1278)	:MTEST:RMODe:FACTion:STOP (see page 590)	
:MTEST:{STARt   STOP} (see page 1279)	:RUN (see page 236) or :STOP (see page 240)	
:MTEST:TRIGger:SOURce (see page 1280)	:TRIGger Commands (see page 1047)	There are various commands for setting the source with different types of triggers.
:PRINT? (see page 1281)	:DISPlay:DATA? (see page 339)	
:SAVE:IMAGe:AREA (see page 1283)	none	
:TIMEbase:DELay (see page 1287)	:TIMEbase:POSIon (see page 1038) or :TIMEbase:WINDOW:POSIon (see page 1043)	TIMEbase:POSIon is position value of main time base; TIMEbase:WINDOW:POSIon is position value of zoomed (delayed) time base window.
:SBUS<n>:LIN:SIGNAl:DEFinition (see page 1284)	none	
:SBUS<n>:SPI:SOURce:DATA (see page 1285)	:SBUS<n>:SPI:SOURce:MOSI (see page 890)	
:SYSTem:MENU (see page 1286)	:DISPlay:MENU (see page 343)	No change in behavior.
:TRIGger:THReShold (see page 1288)	:POD<n>:THReShold (see page 605) or :DIGItal<d>:THReShold (see page 327)	
:TRIGger:TV:TMoDe (see page 1289)	:TRIGger:TV:MODE (see page 1124)	

**Discontinued Commands** Discontinued commands are commands that were used by previous oscilloscopes, but are not supported by the InfiniiVision 3000T X-Series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

Discontinued Command	Current Command Equivalent	Comments
ASTore	:DISPlay:PERSistence INFinite (see <a href="#">page 345</a> )	
CHANnel:MATH	:FUNCTION:OPERation (see <a href="#">page 405</a> )	ADD not included
CHANnel<n>:PROTect	:CHANnel<n>:PROTection (see <a href="#">page 295</a> )	Previous form of this command was used to enable/disable 50Ω protection. The new command resets a tripped protect and the query returns the status of TRIPed or NORMal.
DISPlay:INVerse	none	
DISPlay:COLumn	none	
DISPlay:FREeze	none	
DISPlay:GRID	none	
DISPlay:LINE	none	
DISPlay:PIXel	none	
DISPlay:POSITION	none	
DISPlay:ROW	none	
DISPlay:TEXT	none	
FUNCTION:MOVE	none	
FUNCTION:PEAKs	none	
HARDcopy:ADDRess	none	Only parallel printer port is supported. GPIB printing not supported
MASK	none	All commands discontinued, feature not available
:POWER:SIGNals:CYCLES	:POWER:SIGNals:CYCLES:HARMonics (see <a href="#">page 652</a> ) :POWER:SIGNals:CYCLES:QUALity (see <a href="#">page 653</a> )	This command was separated into several other commands for specific types of power analysis.

Discontinued Command	Current Command Equivalent	Comments
:POWER:SIGNals:DURation	:POWER:SIGNals:DURation:EFFiciency (see <a href="#">page 654</a> ) :POWER:SIGNals:DURation:MODulation (see <a href="#">page 655</a> ) :POWER:SIGNals:DURation:ONOFF:OFF (see <a href="#">page 656</a> ) :POWER:SIGNals:DURation:ONOFF:ON (see <a href="#">page 657</a> ) :POWER:SIGNals:DURation:RIPPLE (see <a href="#">page 658</a> ) :POWER:SIGNals:DURation:TRANSIENT (see <a href="#">page 659</a> )	This command was separated into several other commands for specific types of power analysis.
:POWER:SIGNals:VMAXimum	:POWER:SIGNals:VMAXimum:INRUSH (see <a href="#">page 662</a> ) :POWER:SIGNals:VMAXimum:ONOFF:OFF (see <a href="#">page 663</a> ) :POWER:SIGNals:VMAXimum:ONOFF:ON (see <a href="#">page 664</a> )	This command was separated into several other commands for specific types of power analysis.
:POWER:SIGNals:VSTeady	:POWER:SIGNals:VSTeady:ONOFF:OFF (see <a href="#">page 665</a> ) :POWER:SIGNals:VSTeady:ONOFF:ON (see <a href="#">page 666</a> ) :POWER:SIGNals:VSTeady:TRANSIENT (see <a href="#">page 667</a> )	This command was separated into several other commands for specific types of power analysis.
:POWER:SLEW:VALue	none	Slew rate values are now displayed using max and min measurements of a differentiate math function signal.
:PWRenable	none	The Power Event Enable Register does not exist in the 3000T X-Series oscilloscopes.
:PWRRegister[:EVENT]	none	The Power Event Event Register does not exist in the 3000T X-Series oscilloscopes.
SYSTem:KEY	none	
TEST:ALL	*TST (Self Test) (see <a href="#">page 201</a> )	
TRACE subsystem	none	All commands discontinued, feature not available
TRIGger:ADVanced subsystem		Use new GLITCH, PATTERN, or TV trigger modes

Discontinued Command	Current Command Equivalent	Comments
TRIGger:TV:FIELd	:TRIGger:TV:MODE (see page 1124)	
TRIGger:TV:TVHFrej		
TRIGger:TV:VIR	none	
:TRIGger:USB:SOURce:DMINus	none	USB serial decode and triggering is not supported on the 3000T X-Series oscilloscopes.
:TRIGger:USB:SOURce:DPLus	none	
:TRIGger:USB:SPEEd	none	
:TRIGger:USB:TRIGger	none	
VAUToscale	none	

**Discontinued Parameters** Some previous oscilloscope queries returned control setting values of OFF and ON. The InfiniiVision 3000T X-Series oscilloscopes only return the enumerated values 0 (for off) and 1 (for on).

**:CHANnel:ACTivity****0** (see [page 1334](#))**Command Syntax** :CHANnel:ACTivity

The :CHANnel:ACTivity command clears the cumulative edge variables for the next activity query.

**NOTE** The :CHANnel:ACTivity command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :ACTivity command (see [page 207](#)) instead.

**Query Syntax** :CHANnel:ACTivity?

The :CHANnel:ACTivity? query returns the active edges since the last clear, and returns the current logic levels.

**Return Format** <edges>,<levels><NL>

<edges> ::= presence of edges (32-bit integer in NR1 format).

<levels> ::= logical highs or lows (32-bit integer in NR1 format).

**NOTE** A bit equal to zero indicates that no edges were detected at the specified threshold since the last clear on that channel. Edges may have occurred that were not detected because of the threshold setting.

A bit equal to one indicates that edges have been detected at the specified threshold since the last clear on that channel.

**:CHANnel:LABEL** (see [page 1334](#))**Command Syntax** :CHANnel:LABEL <source\_text><string>

```
<source_text> ::= {CHANnel1 | CHANnel2 | DIGital<d>}  
<d> ::= 0 to (# digital channels - 1) in NR1 format  
<string> ::= quoted ASCII string
```

The :CHANnel:LABEL command sets the source text to the string that follows. Setting a channel will also result in the name being added to the label list.

**NOTE**

The :CHANnel:LABEL command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:LABEL command (see [page 287](#)) or :DIGital<n>:LABEL command (see [page 324](#)).

**Query Syntax** :CHANnel:LABEL?

The :CHANnel:LABEL? query returns the label associated with a particular analog channel.

**Return Format** <string><NL>

```
<string> ::= quoted ASCII string
```

## :CHANnel:THreshold

 (see [page 1334](#))

**Command Syntax**

```
:CHANnel:THreshold <channel group>, <threshold type> [, <value>]
<channel group> ::= {POD1 | POD2}
<threshold type> ::= {CMOS | ECL | TTL | USERdef}
<value> ::= voltage for USERdef in NR3 format [volt_type]
[volt_type] ::= {V | mV (-3) | uV (-6)}
```

The :CHANnel:THreshold command sets the threshold for a group of channels. The threshold is either set to a predefined value or to a user-defined value. For the predefined value, the voltage parameter is ignored.

### NOTE

The :CHANnel:THreshold command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :POD<n>:THreshold command (see [page 605](#)) or :DIGital<n>:THreshold command (see [page 327](#)).

### Query Syntax

```
:CHANnel:THreshold? <channel group>
```

The :CHANnel:THreshold? query returns the voltage and threshold text for a specific group of channels.

### Return Format

```
<threshold type> [, <value>]<NL>
<threshold type> ::= {CMOS | ECL | TTL | USERdef}
<value> ::= voltage for USERdef (float 32 NR3)
```

### NOTE

- CMOS = 2.5V
- TTL = 1.5V
- ECL = -1.3V
- USERdef ::= -6.0V to 6.0V

## :CHANnel2:SKEW

 (see [page 1334](#))

**Command Syntax** :CHANnel2:SKEW <skew value>

<skew value> ::= skew time in NR3 format

<skew value> ::= -100 ns to +100 ns

The :CHANnel2:SKEW command sets the skew between channels 1 and 2. The maximum skew is +/-100 ns. You can use the oscilloscope's analog probe skew control to remove cable delay errors between channel 1 and channel 2.

### NOTE

The :CHANnel2:SKEW command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:PROBe:SKEW command (see [page 293](#)) instead.

### NOTE

This command is only valid for the two channel oscilloscope models.

### Query Syntax

:CHANnel2:SKEW?

The :CHANnel2:SKEW? query returns the current probe skew setting for the selected channel.

### Return Format

<skew value><NL>

<skew value> ::= skew value in NR3 format

### See Also

- "Introduction to :CHANnel<n> Commands" on page 281

**:CHANnel<n>:INPut****0** (see [page 1334](#))**Command Syntax**    `:CHANnel<n>:INPut <impedance>``<impedance> ::= {ONEMeg | FIFTy}``<n> ::= 1 to (# analog channels) in NR1 format`

The :CHANnel<n>:INPut command selects the input impedance setting for the specified channel. The legal values for this command are ONEMeg ( $1 \text{ M}\Omega$ ) and FIFTy ( $50\Omega$ ).

**NOTE**

The :CHANnel<n>:INPut command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:IMPedance command (see [page 285](#)) instead.

**Query Syntax**    `:CHANnel<n>:INPut?`

The :CHANnel<n>:INPut? query returns the current input impedance setting for the specified channel.

**Return Format**    `<impedance value><NL>``<impedance value> ::= {ONEM | FIFT}`

## :CHANnel<n>:PMODe

 (see [page 1334](#))

**Command Syntax** :CHANnel<n>:PMODe <pmode value>

<pmode value> ::= {AUTo | MANual}

<n> ::= 1 to (# analog channels) in NR1 format

The probe sense mode is controlled internally and cannot be set. If a probe with sense is connected to the specified channel, auto sensing is enabled; otherwise, the mode is manual.

If the PMODe sent matches the oscilloscope's setting, the command will be accepted. Otherwise, a setting conflict error is generated.

### NOTE

The :CHANnel<n>:PMODe command is an obsolete command provided for compatibility to previous oscilloscopes.

**Query Syntax** :CHANnel<n>:PMODe?

The :CHANnel<n>:PMODe? query returns AUT if an autosense probe is attached and MAN otherwise.

**Return Format** <pmode value><NL>

<pmode value> ::= {AUT | MAN}

**:DISPlay:CONNect****0** (see [page 1334](#))**Command Syntax**    `:DISPlay:CONNect <connect>``<connect> ::= {{ 1 | ON} | {0 | OFF}}`

The :DISPlay:CONNect command turns vectors on and off. When vectors are turned on, the oscilloscope displays lines connecting sampled data points. When vectors are turned off, only the sampled data is displayed.

**NOTE**

The :DISPlay:CONNect command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISPlay:VECTors command (see [page 346](#)) instead.

**Query Syntax**    `:DISPlay:CONNect?`

The :DISPlay:CONNect? query returns the current state of the vectors setting.

**Return Format**    `<connect><NL>``<connect> ::= {1 | 0}`**See Also**    • " [":DISPlay:VECTors](#)" on page 346

## :DISPlay:ORDer

 (see [page 1334](#))

**Query Syntax** :DISPlay:ORDer?

The :DISPlay:ORDer? query returns a list of digital channel numbers in screen order, from top to bottom, separated by commas. Busing is displayed as digital channels with no separator. For example, in the following list, the bus consists of digital channels 4 and 5: DIG1, DIG4 DIG5, DIG7.

**NOTE**

The :DISPlay:ORDer command is an obsolete command provided for compatibility to previous oscilloscopes. This command is only available on the MSO models.

**Return Format**

```
<order><NL>
<order> ::= Unquoted ASCII string
```

**NOTE**

A return value is included for each digital channel. A return value of NONE indicates that a channel is turned off.

**See Also**

- [":DIGItal<d>:POsition"](#) on page 325

**Example Code**

```
' DISP_ORDER - Set the order the channels are displayed on the
' analyzer. You can enter between 1 and 32 channels at one time.
' If you leave out channels, they will not be displayed.

' Display ONLY channel 0 and channel 10 in that order.
myScope.WriteString ":DISPLAY:ORDER 0,10"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

## :ERASe

 (see [page 1334](#))

**Command Syntax** :ERASe

The :ERASe command erases the screen.

### NOTE

The :ERASe command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISPLAY:CLEAR command (see [page 338](#)) instead.

---

**:EXTernal:PMODE****0** (see [page 1334](#))

**Command Syntax**    `:EXTernal:PMODE <pmode value>`  
`<pmode value> ::= {AUTo | MANual}`

The probe sense mode is controlled internally and cannot be set. If a probe with sense is connected to the specified channel, auto sensing is enabled; otherwise, the mode is manual.

If the pmode sent matches the oscilloscope's setting, the command will be accepted. Otherwise, a setting conflict error is generated.

**NOTE**

The :EXTernal:PMODE command is an obsolete command provided for compatibility to previous oscilloscopes.

**Query Syntax**    `:EXTernal:PMODE?`

The :EXTernal:PMODE? query returns AUT if an autosense probe is attached and MAN otherwise.

**Return Format**    `<pmode value><NL>`  
`<pmode value> ::= {AUT | MAN}`

## :FUNCTION:GOFT:OPERation

 (see [page 1334](#))

**Command Syntax** :FUNCTION:GOFT:OPERation <operation>

<operation> ::= {ADD | SUBTract | MULTiply}

The :FUNCTION:GOFT:OPERation command sets the math operation for the g(t) source that can be used as the input to transform or filter functions (if available):

- ADD – Source1 + source2.
- SUBTract – Source1 - source2.
- MULTiply – Source1 \* source2.

The :FUNCTION:GOFT:SOURce1 and :FUNCTION:GOFT:SOURce2 commands are used to select source1 and source2.

## :FUNCTION:GOFT:SOURce1

 (see [page 1334](#))

**Command Syntax**

```
:FUNCTION:GOFT:SOURce1 <value>
<value> ::= CHANnel<n>
<n> ::= {1 | 2 | 3 | 4} for 4ch models
<n> ::= {1 | 2} for 2ch models
```

The :FUNCTION:GOFT:SOURce1 command selects the first input channel for the g(t) source that can be used as the input to transform or filter functions (if available).

**Query Syntax**

```
:FUNCTION:GOFT:SOURce1?
```

The :FUNCTION:GOFT:SOURce1? query returns the current selection for the first input channel for the g(t) source.

**Return Format**

```
<value><NL>
<value> ::= CHAN<n>
<n> ::= {1 | 2 | 3 | 4} for the 4ch models
<n> ::= {1 | 2} for the 2ch models
```

**See Also**

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":FUNCTION:GOFT:SOURce2](#)" on page 1252
- "[":FUNCTION:GOFT:OPERation](#)" on page 1250

**:FUNCTION:GOFT:SOURce2** (see [page 1334](#))

**Command Syntax**    `:FUNCTION:GOFT:SOURce2 <value>`

`<value> ::= CHANnel<n>`

`<n> ::= {1 | 2 | 3 | 4} for 4ch models`

`<n> ::= {1 | 2} for 2ch models`

The :FUNCTION:GOFT:SOURce2 command selects the second input channel for the g(t) source that can be used as the input to transform or filter functions (if available).

**Query Syntax**    `:FUNCTION:GOFT:SOURce2?`

The :FUNCTION:GOFT:SOURce2? query returns the current selection for the second input channel for the g(t) source.

**Return Format**    `<value><NL>`

`<value> ::= CHAN<n>`

`<n> ::= {1 | 2 | 3 | 4} for 4ch models`

`<n> ::= {1 | 2} for 2ch models`

**See Also**

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[":FUNCTION:GOFT:SOURce1](#)" on page 1251
- "[":FUNCTION:GOFT:OPERation](#)" on page 1250

## :FUNCTION:SOURce

 (see [page 1334](#))

### Command Syntax

```
:FUNCTION:SOURce <value>
<value> ::= {CHANnel<n> | ADD | SUBTract | MULTiply}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :FUNCTION:SOURce command is only used when an FFT (Fast Fourier Transform), DIFF, or INT operation is selected (see the :FUNCTION:OPERation command for more information about selecting an operation). The :FUNCTION:SOURce command selects the source for function operations. Choose CHANnel<n>, or ADD, SUBT, or MULT to specify the desired source for function DIFF (differentiate), INTegrate, and FFT operations specified by the :FUNCTION:OPERation command.

### NOTE

The :FUNCTION:SOURce command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :FUNCTION:SOURce1 command (see [page 413](#)) instead.

### Query Syntax

```
:FUNCTION:SOURce?
```

The :FUNCTION:SOURce? query returns the current source for function operations.

### Return Format

```
<value><NL>
<value> ::= {CHAN<n> | ADD | SUBT | MULT}
<n> ::= 1 to (# analog channels) in NR1 format
```

### See Also

- "[Introduction to :FUNCTION<m> Commands](#)" on page 383
- "[:FUNCTION<m>:OPERation](#)" on page 405

**:FUNCTION:VIEW****0** (see [page 1334](#))

**Command Syntax**    `:FUNCTION:VIEW <view>`  
`<view> ::= {{1 | ON} | (0 | OFF)}`

The :FUNCTION:VIEW command turns the selected function on or off. When ON is selected, the function performs as specified using the other FUNCTION commands. When OFF is selected, function is neither calculated nor displayed.

**NOTE**

The :FUNCTION:VIEW command is provided for backward compatibility to previous oscilloscopes. Use the :FUNCTION:DISPLAY command (see [page 391](#)) instead.

**Query Syntax**    `:FUNCTION:VIEW?`

The :FUNCTION:VIEW? query returns the current state of the selected function.

**Return Format**    `<view><NL>`  
`<view> ::= {1 | 0}`

## :HARDcopy:DESTination

 (see [page 1334](#))

**Command Syntax**    `:HARDcopy:DESTination <destination>`  
`<destination> ::= {CENTronics | FLOPpy}`

The :HARDcopy:DESTination command sets the hardcopy destination.

### NOTE

The :HARDcopy:DESTination command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:FILEname command (see [page 1256](#)) instead.

**Query Syntax**    `:HARDcopy:DESTination?`

The :HARDcopy:DESTination? query returns the selected hardcopy destination.

**Return Format**    `<destination><NL>`  
`<destination> ::= {CENT | FLOP}`

**See Also**    · "Introduction to :HARDcopy Commands" on page 418

**:HARDcopy:FILEname****0** (see [page 1334](#))

**Command Syntax**    `:HARDcopy:FILEname <string>`  
`<string> ::= quoted ASCII string`

The HARDcopy:FILEname command sets the output filename for those print formats whose output is a file.

**NOTE**

The :HARDcopy:FILEname command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :SAVE:FILEname command ([see page 698](#)) and :RECall:FILEname command ([see page 687](#)) instead.

**Query Syntax**    `:HARDcopy:FILEname?`

The :HARDcopy:FILEname? query returns the current hardcopy output filename.

**Return Format**    `<string><NL>`  
`<string> ::= quoted ASCII string`

**See Also**    · "Introduction to :HARDcopy Commands" on page 418

## :HARDcopy:GRAYscale

 (see [page 1334](#))

**Command Syntax**    `:HARDcopy:GRAYscale <gray>`

`<gray> ::= {{OFF | 0} | {ON | 1}}`

The :HARDcopy:GRAYscale command controls whether grayscaling is performed in the hardcopy dump.

### NOTE

The :HARDcopy:GRAYscale command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:PAlette command (see [page 431](#)) instead. ("":HARDcopy:GRAYscale ON" is the same as "":HARDcopy:PAlette GRAYscale" and "":HARDcopy:GRAYscale OFF" is the same as "":HARDcopy:PAlette COLOR".)

**Query Syntax**    `:HARDcopy:GRAYscale?`

The :HARDcopy:GRAYscale? query returns a flag indicating whether grayscaling is performed in the hardcopy dump.

**Return Format**    `<gray><NL>`

`<gray> ::= {0 | 1}`

**See Also**    · ["Introduction to :HARDcopy Commands"](#) on page 418

**:HARDcopy:IGColors** (see [page 1334](#))**Command Syntax**    `:HARDcopy:IGColors <value>``<value> ::= {{OFF | 0} | {ON | 1}}`

The HARDcopy:IGColors command controls whether the graticule colors are inverted or not.

**NOTE**

The :HARDcopy:IGColors command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:INKSaver (see [page 423](#)) command instead.

**Query Syntax**    `:HARDcopy:IGColors?`

The :HARDcopy:IGColors? query returns a flag indicating whether graticule colors are inverted or not.

**Return Format**    `<value><NL>``<value> ::= {0 | 1}`**See Also**    • "Introduction to :HARDcopy Commands" on page 418

## :HARDcopy:PDRiver

 (see [page 1334](#))

**Command Syntax** :HARDcopy:PDRiver <driver>

```
<driver> ::= {AP2XXX | AP21XX | {AP2560 | AP25} | {DJ350 | DJ35} |
    DJ6XX | {DJ630 | DJ63} | DJ6Special | DJ6Photo |
    DJ8Special | DJ8XX | DJ9Vip | OJPRokx50 | DJ9XX | GVIP |
    DJ55XX | {PS470 | PS47} {PS100 | PS10} | CLASer |
    MLASer | LJFastraster | POSTscript}
```

The HARDcopy:PDRiver command sets the hardcopy printer driver used for the selected printer.

If the correct driver for the selected printer can be identified, it will be selected and cannot be changed.

### NOTE

The :HARDcopy:PDRiver command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:APRinter (see [page 420](#)) command instead.

**Query Syntax** :HARDcopy:PDRiver?

The :HARDcopy:PDRiver? query returns the selected hardcopy printer driver.

**Return Format** <driver><NL>

```
<driver> ::= {AP2X | AP21 | AP25 | DJ35 | DJ6 | DJ63 | DJ6S | DJ6P |
    DJ8S | DJ8 | DJ9V | OJPR | DJ9 | GVIP | DJ55 | PS10 |
    PS47 | CLAS | MLAS | LJF | POST}
```

**See Also** • "Introduction to :HARDcopy Commands" on page 418

## :MEASure:LOWER

 (see [page 1334](#))

**Command Syntax**    `:MEASure:LOWER <voltage>`

The :MEASure:LOWER command sets the lower measurement threshold value. This value and the UPPer value represent absolute values when the thresholds are ABSolute and percentage when the thresholds are PERCent as defined by the :MEASure:DEFine THResholds command.

### NOTE

The :MEASure:LOWER command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THResholds command (see [page 485](#)) instead.

**Query Syntax**    `:MEASure:LOWER?`

The :MEASure:LOWER? query returns the current lower threshold level.

**Return Format**    `<voltage><NL>`

`<voltage>` ::= the user-defined lower threshold in volts in NR3 format

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:THResholds](#)" on page 1263
- "[":MEASure:UPPer](#)" on page 1269

## :MEASure:SCRatch

 (see [page 1334](#))

**Command Syntax** :MEASure:SCRatch

The :MEASure:SCRatch command clears all selected measurements and markers from the screen.

### NOTE

The :MEASure:SCRatch command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:CLEar command (see [page 482](#)) instead.

---

**:MEASure:TDELta**

 (see [page 1334](#))

**Query Syntax** `:MEASure:TDELta?`

The `:MEASure:TDELta?` query returns the time difference between the Tstop marker (X2 cursor) and the Tstart marker (X1 cursor).

$$\text{Tdelta} = \text{Tstop} - \text{Tstart}$$

Tstart is the time at the start marker (X1 cursor) and Tstop is the time at the stop marker (X2 cursor). No measurement is made when the `:MEASure:TDELta?` query is received by the oscilloscope. The delta time value that is output is the current value. This is the same value as the front-panel cursors delta X value.

**NOTE**

The `:MEASure:TDELta` command is an obsolete command provided for compatibility to previous oscilloscopes. Use the `:MARKer:XDELta` command (see [page 450](#)) instead.

**Return Format**

`<value><NL>`

`<value>` ::= time difference between start and stop markers in NR3 format

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MARKer:X1Position](#)" on page 445
- "[:MARKer:X2Position](#)" on page 448
- "[:MARKer:XDELta](#)" on page 450
- "[:MEASure:TSTArt](#)" on page 1266
- "[:MEASure:TSTOP](#)" on page 1267

## :MEASure:THResholds

 (see [page 1334](#))

**Command Syntax** :MEASure:THResholds {T1090 | T2080 | VOLTage}

The :MEASure:THResholds command selects the thresholds used when making time measurements.

### NOTE

The :MEASure:THResholds command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THResholds command (see [page 485](#)) instead.

**Query Syntax** :MEASure:THResholds?

The :MEASure:THResholds? query returns the current thresholds selected when making time measurements.

**Return Format** {T1090 | T2080 | VOLTage}<NL>

{T1090} uses the 10% and 90% levels of the selected waveform.

{T2080} uses the 20% and 80% levels of the selected waveform.

{VOLTage} uses the upper and lower voltage thresholds set by the UPPer and LOWER commands on the selected waveform.

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:LOWER](#)" on page 1260
- "[":MEASure:UPPer](#)" on page 1269

## :MEASure:TMAX

 (see [page 1334](#))

**Command Syntax**    `:MEASure:TMAX [<source>]`

```
<source> ::= {CHANnel<n> | FUNCTION | MATH}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:TMAX command installs a screen measurement and starts an X-at-Max-Y measurement on the selected waveform. If the optional source is specified, the current source is modified.

### NOTE

The :MEASure:TMAX command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:XMAX command ([see page 540](#)) instead.

**Query Syntax**    `:MEASure:TMAX? [<source>]`

The :MEASure:TMAX? query returns the horizontal axis value at which the maximum vertical value occurs on the current source. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

**Return Format**    `<value><NL>`

```
<value> ::= time at maximum in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:TMIN](#)" on page 1265
- "[":MEASure:XMAX](#)" on page 540
- "[":MEASure:XMIN](#)" on page 541

## :MEASure:TMIN

 (see [page 1334](#))

**Command Syntax**    `:MEASure:TMIN [<source>]`

```
<source> ::= {CHANnel<n> | FUNCtion | MATH}
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:TMIN command installs a screen measurement and starts an X-at-Min-Y measurement on the selected waveform. If the optional source is specified, the current source is modified.

### NOTE

The :MEASure:TMIN command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:XMIN command (see [page 541](#)) instead.

**Query Syntax**    `:MEASure:TMIN? [<source>]`

The :MEASure:TMIN? query returns the horizontal axis value at which the minimum vertical value occurs on the current source. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

**Return Format**    `<value><NL>`

```
<value> ::= time at minimum in NR3 format
```

**See Also**

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:TMAX"](#) on page 1264
- "[":MEASure:XMAX"](#) on page 540
- "[":MEASure:XMIN"](#) on page 541

**:MEASure:TSTArt**

 (see [page 1334](#))

**Command Syntax**    `:MEASure:TSTArt <value> [suffix]`

`<value>` ::= time at the start marker in seconds

`[suffix]` ::= {s | ms | us | ns | ps}

The :MEASure:TSTArt command moves the start marker (X1 cursor) to the specified time with respect to the trigger time.

**NOTE**

The short form of this command, TSTA, does not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)). The normal short form "TST" would be the same for both TSTArt and TSTOp, so sending TST for the TSTArt command produces an error.

**NOTE**

The :MEASure:TSTArt command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:X1Position command (see [page 445](#)) instead.

**Query Syntax**

`:MEASure:TSTArt?`

The :MEASure:TSTArt? query returns the time at the start marker (X1 cursor).

**Return Format**

`<value><NL>`

`<value>` ::= time at the start marker in NR3 format

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MARKer:X1Position](#)" on page 445
- "[:MARKer:X2Position](#)" on page 448
- "[:MARKer:XDELta](#)" on page 450
- "[:MEASure:TDELta](#)" on page 1262
- "[:MEASure:TSTOP](#)" on page 1267

## :MEASure:TSTOp

 (see [page 1334](#))

**Command Syntax**    `:MEASure:TSTOp <value> [suffix]`

`<value>` ::= time at the stop marker in seconds

`[suffix]` ::= {s | ms | us | ns | ps}

The :MEASure:TSTOp command moves the stop marker (X2 cursor) to the specified time with respect to the trigger time.

### NOTE

The short form of this command, TSTO, does not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)). The normal short form "TST" would be the same for both TSTArt and TSTOp, so sending TST for the TSTOp command produces an error.

### NOTE

The :MEASure:TSTOp command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:X2Position command (see [page 448](#)) instead.

#### Query Syntax

`:MEASure:TSTOp?`

The :MEASure:TSTOp? query returns the time at the stop marker (X2 cursor).

#### Return Format

`<value><NL>`

`<value>` ::= time at the stop marker in NR3 format

#### See Also

- "[Introduction to :MARKer Commands](#)" on page 441
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MARKer:X1Position](#)" on page 445
- "[:MARKer:X2Position](#)" on page 448
- "[:MARKer:XDELta](#)" on page 450
- "[:MEASure:TDELta](#)" on page 1262
- "[:MEASure:TSTArt](#)" on page 1266

## :MEASure:TVOLt

**0** (see [page 1334](#))

**Query Syntax**

```
:MEASure:TVOLt? <value>, [<slope>]<occurrence>[,<source>]

<value> ::= the voltage level that the waveform must cross.

<slope> ::= direction of the waveform. A rising slope is indicated by
            a plus sign (+). A falling edge is indicated by a minus
            sign (-).

<occurrence> ::= the transition to be reported. If the occurrence
                  number is one, the first crossing is reported. If
                  the number is two, the second crossing is reported,
                  etc.

<source> ::= {<digital channels> | CHANnel<n> | FUNCTion | MATH}

<digital channels> ::= {DIGItal<d>} for the MSO models

<n> ::= 1 to (# analog channels) in NR1 format

<d> ::= 0 to (# digital channels - 1) in NR1 format
```

When the :MEASure:TVOLt? query is sent, the displayed signal is searched for the specified voltage level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to the query.

The specified voltage can be negative or positive. To specify a negative voltage, use a minus sign (-). The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of the occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the specified voltage level in the positive direction. Once this voltage crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage for the specified number of times in the direction specified.

If the optional source parameter is specified, the current source is modified.

### NOTE

The :MEASure:TVOLt command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:TVALue command (see [page 527](#)).

### Return Format

```
<value><NL>

<value> ::= time in seconds of the specified voltage crossing
            in NR3 format
```

## :MEASure:UPPer

 (see [page 1334](#))

### Command Syntax

`:MEASure:UPPer <value>`

The :MEASure:UPPer command sets the upper measurement threshold value. This value and the LOWER value represent absolute values when the thresholds are ABSolute and percentage when the thresholds are PERCent as defined by the :MEASure:DEFine THReholds command.

### NOTE

The :MEASure:UPPer command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THReholds command (see [page 485](#)) instead.

### Query Syntax

`:MEASure:UPPer?`

The :MEASure:UPPer? query returns the current upper threshold level.

### Return Format

`<value><NL>`

`<value>` ::= the user-defined upper threshold in NR3 format

### See Also

- "[Introduction to :MEASure Commands](#)" on page 476
- "[":MEASure:LOWER"](#) on page 1260
- "[":MEASure:THReholds"](#) on page 1263

**:MEASure:VDELta**

 (see [page 1334](#))

**Query Syntax**    `:MEASure:VDELta?`

The `:MEASure:VDELta?` query returns the voltage difference between vertical marker 1 (Y1 cursor) and vertical marker 2 (Y2 cursor). No measurement is made when the `:MEASure:VDELta?` query is received by the oscilloscope. The delta value that is returned is the current value. This is the same value as the front-panel cursors delta Y value.

$VDELta = \text{value at marker 2} - \text{value at marker 1}$

**NOTE**

The `:MEASure:VDELta` command is an obsolete command provided for compatibility to previous oscilloscopes. Use the `:MARKer:YDELta` command ([see page 457](#)) instead.

**Return Format**    `<value><NL>`

`<value>` ::= delta V value in NR1 format

**See Also**

- "[Introduction to :MARKer Commands](#)" on page 441
- "[Introduction to :MEASure Commands](#)" on page 476
- "[:MARKer:Y1Position](#)" on page 454
- "[:MARKer:Y2Position](#)" on page 456
- "[:MARKer:YDELta](#)" on page 457
- "[:MEASure:TDELta](#)" on page 1262
- "[:MEASure:TSTArt](#)" on page 1266

## :MEASure:VSTArt

 (see [page 1334](#))

**Command Syntax**    `:MEASure:VSTArt <vstart_argument>`

`<vstart_argument> ::= value for vertical marker 1`

The :MEASure:VSTArt command moves the vertical marker (Y1 cursor) to the specified value corresponding to the selected source. The source can be selected by the MARKer:X1Y1source command.

### NOTE

The short form of this command, VSTA, does not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)). The normal short form, VST, would be the same for both VSTArt and VSTOp, so sending VST for the VSTArt command produces an error.

### NOTE

The :MEASure:VSTArt command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:Y1Position command (see [page 454](#)) instead.

#### Query Syntax

`:MEASure:VSTArt?`

The :MEASure:VSTArt? query returns the current value of the Y1 cursor.

#### Return Format

`<value><NL>`

`<value> ::= voltage at voltage marker 1 in NR3 format`

#### See Also

- ["Introduction to :MARKer Commands"](#) on page 441
- ["Introduction to :MEASure Commands"](#) on page 476
- [":MARKer:Y1Position"](#) on page 454
- [":MARKer:Y2Position"](#) on page 456
- [":MARKer:YDELta"](#) on page 457
- [":MARKer:X1Y1source"](#) on page 446
- [":MEASure:SOURce"](#) on page 517
- [":MEASure:TDELta"](#) on page 1262
- [":MEASure:TSTArt"](#) on page 1266

## :MEASure:VSTOp

 (see [page 1334](#))

**Command Syntax**    `:MEASure:VSTOp <vstop_argument>`

`<vstop_argument> ::= value for Y2 cursor`

The :MEASure:VSTOp command moves the vertical marker 2 (Y2 cursor) to the specified value corresponding to the selected source. The source can be selected by the MARKer:X2Y2source command.

### NOTE

The short form of this command, VSTO, does not follow the defined Long Form to Short Form Truncation Rules (see [page 1336](#)). The normal short form, VST, would be the same for both VSTArt and VSTOp, so sending VST for the VSTOp command produces an error.

### NOTE

The :MEASure:VSTOp command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:Y2Position command (see [page 456](#)) instead.

#### Query Syntax

`:MEASure:VSTOp?`

The :MEASure:VSTOp? query returns the current value of the Y2 cursor.

#### Return Format

`<value><NL>`

`<value> ::= value of the Y2 cursor in NR3 format`

#### See Also

- ["Introduction to :MARKer Commands"](#) on page 441
- ["Introduction to :MEASure Commands"](#) on page 476
- [":MARKer:Y1Position"](#) on page 454
- [":MARKer:Y2Position"](#) on page 456
- [":MARKer:YDELta"](#) on page 457
- [":MARKer:X2Y2source"](#) on page 449
- [":MEASure:SOURce"](#) on page 517
- [":MEASure:TDELta"](#) on page 1262
- [":MEASure:TSTArt"](#) on page 1266

**:MTEST:AMASK:{SAVE | STORe}** (see [page 1334](#))**Command Syntax** :MTEST:AMASK:{SAVE | STORe} "<filename>"

The :MTEST:AMASK:SAVE command saves the automask generated mask to a file. If an automask has not been generated, an error occurs.

The <filename> parameter is an MS-DOS compatible name of the file, a maximum of 254 characters long (including the path name, if used). The filename assumes the present working directory if a path does not precede the file name.

**NOTE**

The :MTEST:AMASK:{SAVE | STORe} command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :SAVE:MASK[:STARt] command (see [page 705](#)) instead.

**See Also** • ["Introduction to :MTEST Commands"](#) on page 569

## :MTEST:AVERage

 (see [page 1334](#))

**Command Syntax**    `:MTEST:AVERage <on_off>`

`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:AVERage command enables or disables averaging. When ON, the oscilloscope acquires multiple data values for each time bucket, and averages them. When OFF, averaging is disabled. To set the number of averages, use the :MTEST:AVERage:COUNt command described next.

### NOTE

The :MTEST:AVERage command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :ACQuire:TYPE AVERage command (see [page 255](#)) instead.

**Query Syntax**    `:MTEST:AVERage?`

The :MTEST:AVERage? query returns the current setting for averaging.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:AVERage:COUNt](#)" on page 1275

## :MTEST:AVERage:COUNt

 (see [page 1334](#))

**Command Syntax** :MTEST:AVERage:COUNt <count>

<count> ::= an integer from 2 to 65536 in NR1 format

The :MTEST:AVERage:COUNt command sets the number of averages for the waveforms. With the AVERage acquisition type, the :MTEST:AVERage:COUNt command specifies the number of data values to be averaged for each time bucket before the acquisition is considered complete for that time bucket.

### NOTE

The :MTEST:AVERage:COUNt command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :ACQuire:COUNt command (see [page 246](#)) instead.

**Query Syntax** :MTEST:AVERage:COUNt?

The :MTEST:AVERage:COUNt? query returns the currently selected count value.

**Return Format** <count><NL>

<count> ::= an integer from 2 to 65536 in NR1 format

**See Also**

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:AVERage"](#) on page 1274

## :MTEST:LOAD

 (see [page 1334](#))

**Command Syntax** :MTEST:LOAD "<filename>"

The :MTEST:LOAD command loads the specified mask file.

The <filename> parameter is an MS-DOS compatible name of the file, a maximum of 254 characters long (including the path name, if used).

### NOTE

The :MTEST:LOAD command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :RECALL:MASK[:START] command (see [page 689](#)) instead.

### See Also

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:AMASK:{SAVE | STORe}](#)" on page 1273

## :MTEST:RUMode

 (see [page 1334](#))

**Command Syntax**

```
:MTEST:RUMode {FORever | TIME,<seconds> | {WAVeforms,<wfm_count>}}
```

<seconds> ::= from 1 to 86400 in NR3 format

<wfm\_count> ::= number of waveforms in NR1 format  
from 1 to 1,000,000,000

The :MTEST:RUMode command determines the termination conditions for the mask test. The choices are FORever, TIME, or WAVeforms.

- FORever – runs the Mask Test until the test is turned off.
- TIME – sets the amount of time in seconds that a mask test will run before it terminates. The <seconds> parameter is a real number from 1 to 86400 seconds.
- WAVeforms – sets the maximum number of waveforms that are required before the mask test terminates. The <wfm\_count> parameter indicates the number of waveforms that are to be acquired; it is an integer from 1 to 1,000,000,000.

### NOTE

The :MTEST:RUMode command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MTEST:RMODe command (see [page 586](#)) instead.

### Query Syntax

```
:MTEST:RUMode?
```

The :MTEST:RUMode? query returns the currently selected termination condition and value.

### Return Format

```
{FOR | TIME,<seconds> | {WAV,<wfm_count>}}<NL>
```

<seconds> ::= from 1 to 86400 in NR3 format

<wfm\_count> ::= number of waveforms in NR1 format  
from 1 to 1,000,000,000

### See Also

- "[Introduction to :MTEST Commands](#)" on page 569
- "[":MTEST:RUMode:SOFailure](#)" on page 1278

**:MTEST:RUMode:SOFailure** (see [page 1334](#))

**Command Syntax**    `:MTEST :RUMode :SOFailure <on_off>`  
`<on_off> ::= {{1 | ON} | {0 | OFF}}`

The :MTEST:RUMode:SOFailure command enables or disables the Stop On Failure run until criteria. When a mask test is run and a mask violation is detected, the mask test is stopped and the acquisition system is stopped.

**NOTE**

The :MTEST:RUMode:SOFailure command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MTEST:RMODe:FACtion:STOP command (see [page 590](#)) instead.

**Query Syntax**    `:MTEST :RUMode :SOFailure?`

The :MTEST:RUMode:SOFailure? query returns the current state of the Stop on Failure control.

**Return Format**    `<on_off><NL>`

`<on_off> ::= {1 | 0}`

**See Also**

- "Introduction to :MTEST Commands" on page 569
- "[:MTEST:RUMode](#)" on page 1277

**:MTEST:{STARt | STOP}** (see [page 1334](#))**Command Syntax** :MTEST:{START | STOP}

The :MTEST:{STARt | STOP} command starts or stops the acquisition system.

**NOTE**

The :MTEST:STARt and :MTEST:STOP commands are obsolete and are provided for backward compatibility to previous oscilloscopes. Use the :RUN command (see [page 236](#)) and :STOP command (see [page 240](#)) instead.

**See Also** • ["Introduction to :MTEST Commands"](#) on page 569

**:MTEST:TRIGger:SOURce** (see [page 1334](#))**Command Syntax**    `:MTEST:TRIGger:SOURce <source>`    `<source> ::= CHANnel<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`

The :MTEST:TRIGger:SOURce command sets the channel to use as the trigger.

**NOTE**

The :MTEST:TRIGger:SOURce command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the trigger source commands (see [page 1047](#)) instead.

**Query Syntax**    `:MTEST:TRIGger:SOURce?`

The :MTEST:TRIGger:SOURce? query returns the currently selected trigger source.

**Return Format**    `<source> ::= CHAN<n>`    `<n> ::= 1 to (# analog channels) in NR1 format`**See Also**

- ["Introduction to :MTEST Commands"](#) on page 569

## :PRINt?

 (see [page 1334](#))

**Query Syntax** :PRINt? [<options>]

```
<options> ::= [<print option>] [,...,<print option>]
<print option> ::= {COLOR | GRAYscale | BMP8bit | BMP}
```

The :PRINt? query pulls image data back over the bus for storage.

### NOTE

The :PRINT command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISPLAY:DATA command (see [page 339](#)) instead.

Print Option	:PRINt command	:PRINt? query	Query Default
COLOR	Sets palette=COLOR		
GRAYscale	Sets palette=GRAYscale		palette=COLOR
PRINTER0,1	Causes the USB printer #0,1 to be selected as destination (if connected)	Not used	N/A
BMP8bit	Sets print format to 8-bit BMP	Selects 8-bit BMP formatting for query	N/A
BMP	Sets print format to BMP	Selects BMP formatting for query	N/A
FACTors	Selects outputting of additional settings information for :PRINT	Not used	N/A
NOFACTORS	Deselects outputting of additional settings information for :PRINT	Not used	N/A

Old Print Option:	Is Now:
Hires	COLOR
Lores	GRAYscale
Parallel	PRINTER0
DISK	invalid
PCL	invalid

**NOTE**

The PRINt? query is not a core command.

---

**See Also**

- "[Introduction to Root \(:\)](#) [Commands](#)" on page 206
- "[Introduction to :HARDcopy Commands](#)" on page 418
- "[:HARDcopy:FACTors](#)" on page 421
- "[:HARDcopy:GRAYscale](#)" on page 1257
- "[:DISPlay:DATA](#)" on page 339

**:SAVE:IMAGe:AREA** (see [page 1334](#))**Query Syntax**    `:SAVE:IMAGe:AREA?`

The `:SAVE:IMAGe:AREA?` query returns the selected image area.

When saving images, this query returns SCR (screen). When saving setups or waveform data, this query returns GRAT (graticule) even though graticule images are not saved.

**Return Format**    `<area><NL>``<area> ::= {GRAT | SCR}`**See Also**

- ["Introduction to :SAVE Commands"](#) on page 695

- [":SAVE:IMAGe\[:STARt\]"](#) on page 699
- [":SAVE:IMAGe:FACTors"](#) on page 700
- [":SAVE:IMAGe:FORMAT"](#) on page 701
- [":SAVE:IMAGe:INKSaver"](#) on page 702
- [":SAVE:IMAGe:PALETTE"](#) on page 703

**:SBUS<n>:LIN:SIGNAl:DEFinition**

 (see [page 1334](#))

**Command Syntax**    `:SBUS<n>:LIN:SIGNAl:DEFinition <value>`  
`<value> ::= {LIN | RX | TX}`

The :SBUS<n>:LIN:SIGNAl:DEFinition command sets the LIN signal type. These signals can be set to:

Dominant low signals:

- LIN – the actual LIN single-end bus signal line.
- RX – the Receive signal from the LIN bus transceiver.
- TX – the Transmit signal to the LIN bus transceiver.

**NOTE**

This command is available, but the only legal value is LIN.

**Query Syntax**    `:SBUS<n>:LIN:SIGNAl:DEFinition?`

The :SBUS<n>:LIN:SIGNAl:DEFinition? query returns the current LIN signal type.

**Return Format**    `<value><NL>`  
`<value> ::= LIN`

**See Also**

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[":TRIGger:MODE"](#) on page 1056
- "[":SBUS<n>:LIN:SIGNAl:BAUDrate"](#) on page 828
- "[":SBUS<n>:LIN:SOURce"](#) on page 829

## :SBUS<n>:SPI:SOURce:DATA

 (see [page 1334](#))

### Command Syntax

```
:SBUS<n>:SPI:SOURce:DATA <source>
<source> ::= {CHANnel<n> | EXTERNAL} for the DSO models
<source> ::= {CHANnel<n> | DIGItal<d>} for the MSO models
<n> ::= 1 to (# analog channels) in NR1 format
<d> ::= 0 to (# digital channels - 1) in NR1 format
```

The :SBUS<n>:SPI:SOURce:DATA command sets the source for the SPI serial MOSI data.

This command is the same as the :SBUS<n>:SPI:SOURce:MOSI command.

### Query Syntax

```
:SBUS<n>:SPI:SOURce:DATA?
```

The :SBUS<n>:SPI:SOURce:DATA? query returns the current source for the SPI serial MOSI data.

### Return Format

```
<source><NL>
```

### See Also

- "[Introduction to :TRIGger Commands](#)" on page 1047
- "[:SBUS<n>:SPI:SOURce:MOSI](#)" on page 890
- "[:SBUS<n>:SPI:SOURce:MISO](#)" on page 889
- "[:SBUS<n>:SPI:SOURce:CLOCK](#)" on page 887
- "[:SBUS<n>:SPI:SOURce:FRAMe](#)" on page 888
- "[:SBUS<n>:SPI:TRIGger:PATTern:MISO:DATA](#)" on page 891
- "[:SBUS<n>:SPI:TRIGger:PATTern:MOSI:DATA](#)" on page 893
- "[:SBUS<n>:SPI:TRIGger:PATTern:MISO:WIDTh](#)" on page 892
- "[:SBUS<n>:SPI:TRIGger:PATTern:MOSI:WIDTh](#)" on page 894

:SYSTem:MENU

 (see [page 1334](#))

**Command Syntax** :SYSTem:MENU <menu>

<menu> ::= {MASK | MEASure | SEGmented | LISTer | POWer}

The :SYSTem:MENU command changes the front panel softkey menu.

## :TIMEbase:DElay

**0** (see [page 1334](#))

**Command Syntax** :TIMEbase:DElay <delay\_value>  
 <delay\_value> ::= time in seconds from trigger to the delay reference point on the screen.

The valid range for delay settings depends on the time/division setting for the main time base.

The :TIMEbase:DElay command sets the main time base delay. This delay is the time between the trigger event and the delay reference point on the screen. The delay reference point is set with the :TIMEbase:REFerence command (see [page 1040](#)).

### NOTE

The :TIMEbase:DElay command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :TIMEbase:POSition command (see [page 1038](#)) instead.

**Query Syntax** :TIMEbase:DElay?

The :TIMEbase:DElay query returns the current delay value.

**Return Format** <delay\_value><NL>

<delay\_value> ::= time from trigger to display reference in seconds in NR3 format.

**Example Code**

```
' TIMEBASE_DELAY - Sets the time base delay. This delay
' is the internal time between the trigger event and the
' onscreen delay reference point.

' Set time base delay to 0.0.
myScope.WriteString ":TIMEBASE:DELAY 0.0"
```

See complete example programs at: [Chapter 42](#), “Programming Examples,” starting on page 1343

**:TRIGger:THreshold**

 (see [page 1334](#))

**Command Syntax**    `:TRIGger:THreshold <channel group>, <threshold type> [, <value>]`  
`<channel group> ::= {POD1 | POD2}`  
`<threshold type> ::= {CMOS | ECL | TTL | USERdef}`  
`<value> ::= voltage for USERdef (floating-point number) [Volt type]`  
`[Volt type] ::= {V | mV | uV}`

The :TRIGger:THreshold command sets the threshold (trigger level) for a pod of 8 digital channels (either digital channels 0 through 7 or 8 through 15). The threshold can be set to a predefined value or to a user-defined value. For the predefined value, the voltage parameter is not required.

**NOTE**

This command is only available on the MSO models.

**NOTE**

The :TRIGger:THreshold command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :POD<n>:THreshold command (see [page 605](#)), :DIGItal<d>:THreshold command (see [page 327](#)), or :TRIGger[:EDGE]:LEVel command (see [page 1073](#)).

**Query Syntax**

`:TRIGger:THreshold? <channel group>`

The :TRIGger:THreshold? query returns the voltage and threshold text for analog channel 1 or 2, or POD1 or POD2.

**Return Format**

```

<threshold type>[, <value>]<NL>
<threshold type> ::= {CMOS | ECL | TTL | USER}
CMOS ::= 2.5V
TTL ::= 1.5V
ECL ::= -1.3V
USERdef ::= range from -8.0V to +8.0V.
<value> ::= voltage for USERdef (a floating-point number in NR1.

```

## :TRIGger:TV:TMode

**0** (see [page 1334](#))

**Command Syntax** :TRIGger:TV:TMode <mode>

```
<mode> ::= {FIELD1 | FIELD2 | AFIELDS | ALINES | LINE | VERTical
             | LFIELD1 | LFIELD2 | LALTernate | LVERTical}
```

The :TRIGger:TV:MODE command selects the TV trigger mode and field. The LVERTical parameter is only available when :TRIGger:TV:STANDARD is GENeric. The LALTernate parameter is not available when :TRIGger:TV:STANDARD is GENeric (see [page 1127](#)).

Old forms for <mode> are accepted:

<mode>	Old Forms Accepted
FIELD1	F1
FIELD2	F2
AFIELD	ALLFIELDS, ALLFLDS
ALINES	ALLLINES
LFIELD1	LINEF1, LINEFIELD1
LFIELD2	LINEF2, LINEFIELD2
LALTernate	LINEALT
LVERTical	LINEVert

### NOTE

The :TRIGger:TV:TMODE command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :TRIGger:TV:MODE command (see [page 1124](#)) instead.

**Query Syntax**

:TRIGger:TV:TMODE?

The :TRIGger:TV:TMODE? query returns the TV trigger mode.

**Return Format**

<value><NL>

```
<value> ::= {FIE1 | FIE2 | AFI | ALIN | LINE | VERT | LFI1 | LFI2
             | LALT | LVER}
```



## 38 Error Messages

-440, Query UNTERMINATED after indefinite response

-430, Query DEADLOCKED

-420, Query UNTERMINATED

-410, Query INTERRUPTED

-400, Query error

-340, Calibration failed

-330, Self-test failed

-321, Out of memory

-320, Storage fault

-315, Configuration memory lost

-314, Save/recall memory lost

-313, Calibration memory lost

-311, Memory error

-310, System error

-300, Device specific error

-278, Macro header not found

-277, Macro redefinition not allowed

-276, Macro recursion error

-273, Illegal macro label

-272, Macro execution error

-258, Media protected

-257, File name error

-256, File name not found

-255, Directory full

-254, Media full

-253, Corrupt media

-252, Missing media

-251, Missing mass storage

-250, Mass storage error

-241, Hardware missing

This message can occur when a feature is unavailable or unlicensed.

For example, serial bus decode commands (which require a four-channel oscilloscope) are unavailable on two-channel oscilloscopes, and some serial bus decode commands are only available on four-channel oscilloscopes when the AMS (automotive serial decode) or LSS (low-speed serial decode) options are licensed.

-240, Hardware error

-231, Data questionable

-230, Data corrupt or stale

-224, Illegal parameter value

-223, Too much data

-222, Data out of range

-221, Settings conflict

-220, Parameter error

-200, Execution error

-183, Invalid inside macro definition

-181, Invalid outside macro definition

-178, Expression data not allowed

-171, Invalid expression

-170, Expression error

-168, Block data not allowed

-161, Invalid block data

-158, String data not allowed

-151, Invalid string data

-150, String data error

-148, Character data not allowed

-138, Suffix not allowed

-134, Suffix too long

-131, Invalid suffix

-128, Numeric data not allowed

-124, Too many digits

-123, Exponent too large

-121, Invalid character in number

-120, Numeric data error

-114, Header suffix out of range

-113, Undefined header

-112, Program mnemonic too long

-109, Missing parameter

-108, Parameter not allowed

-105, GET not allowed

-104, Data type error

-103, Invalid separator

-102, Syntax error

-101, Invalid character

-100, Command error

+10, Software Fault Occurred

+100, File Exists

+101, End-Of-File Found

+102, Read Error

+103, Write Error

+104, Illegal Operation

+105, Print Canceled

+106, Print Initialization Failed

+107, Invalid Trace File

+108, Compression Error

+109, No Data For Operation

A remote operation wants some information, but there is no information available. For example, you may request a stored TIFF image using the :DISPLAY:DATA? query, but there may be no image stored.

+112, Unknown File Type

+113, Directory Not Supported

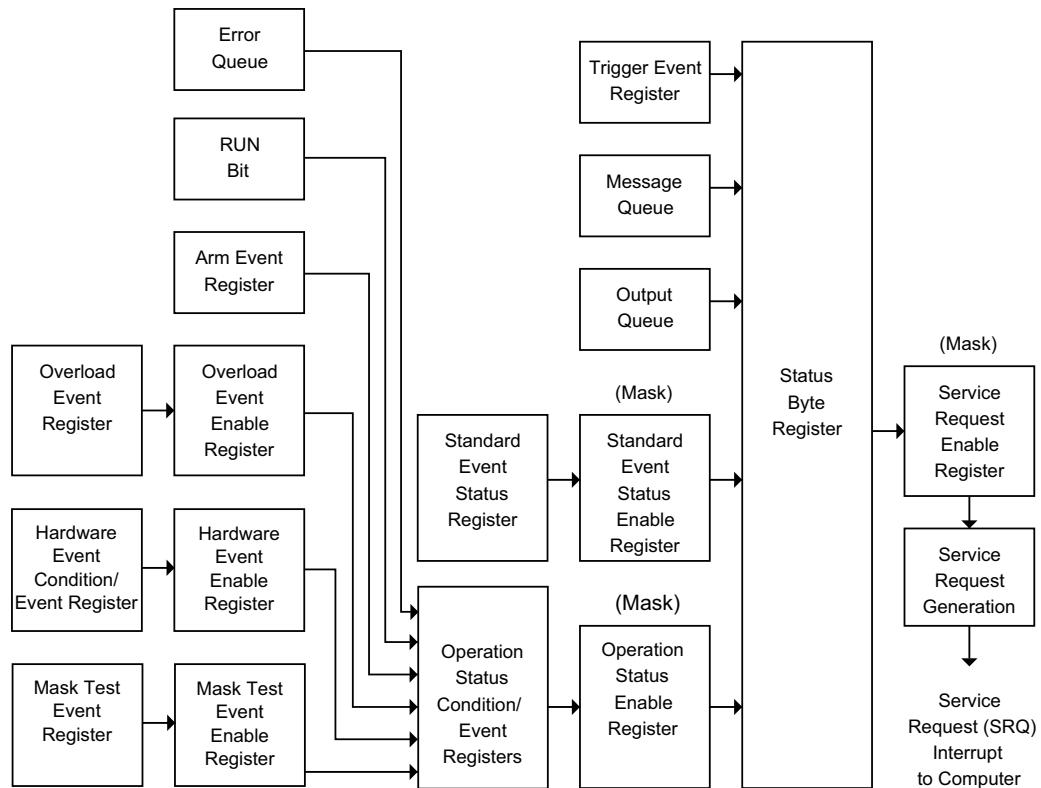


# 39 Status Reporting

Status Reporting Data Structures /	1301
Status Byte Register (STB) /	1304
Service Request Enable Register (SRE) /	1306
Trigger Event Register (TER) /	1307
Output Queue /	1308
Message Queue /	1309
(Standard) Event Status Register (ESR) /	1310
(Standard) Event Status Enable Register (ESE) /	1311
Error Queue /	1312
Operation Status Event Register (:OPERegister[:EVENT]) /	1313
Operation Status Condition Register (:OPERegister:CONDition) /	1314
Arm Event Register (AER) /	1315
Overload Event Register (:OVLRegister) /	1316
Hardware Event Event Register (:HWERegister[:EVENT]) /	1317
Hardware Event Condition Register (:HWERegister:CONDition) /	1318
Mask Test Event Event Register (:MTERegister[:EVENT]) /	1319
Clearing Registers and Queues /	1320
Status Reporting Decision Chart /	1321

IEEE 488.2 defines data structures, commands, and common bit definitions for status reporting (for example, the Status Byte Register and the Standard Event Status Register). There are also instrument-defined structures and bits (for example, the Operation Status Event Register and the Overload Event Register).

An overview of the oscilloscope's status reporting structure is shown in the following block diagram. The status reporting structure allows monitoring specified events in the oscilloscope. The ability to monitor and report these events allows determination of such things as the status of an operation, the availability and reliability of the measured data, and more.



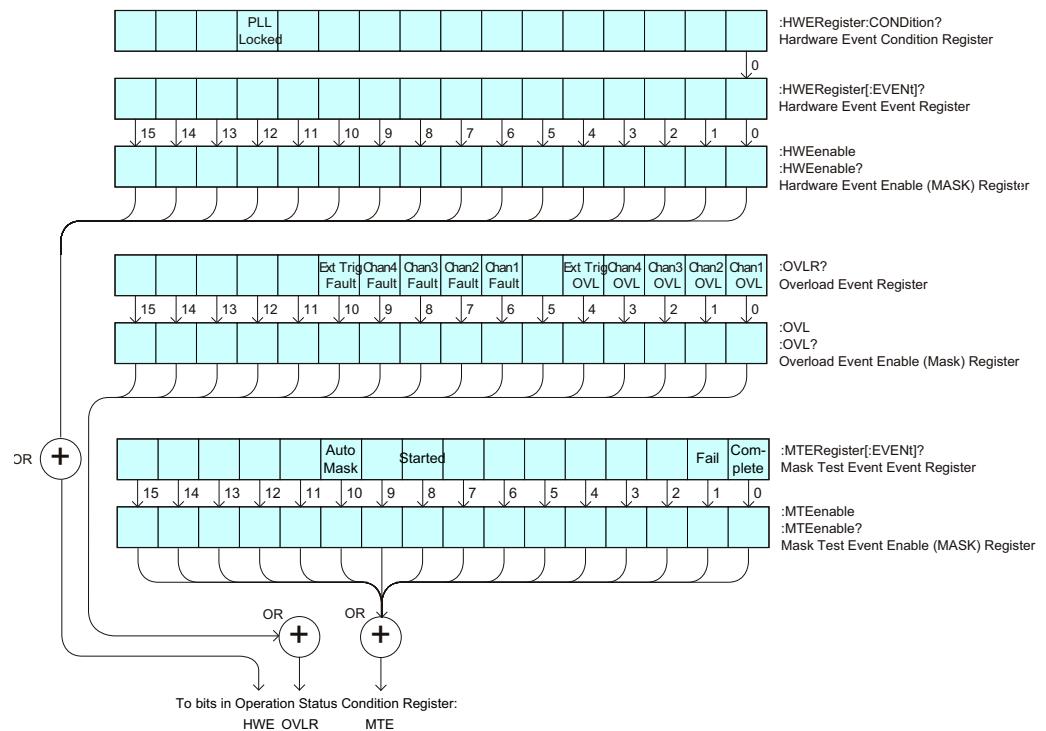
- To monitor an event, first clear the event; then, enable the event. All of the events are cleared when you initialize the instrument.
- To allow a service request (SRQ) interrupt to an external controller, enable at least one bit in the Status Byte Register (by setting, or unmasking, the bit in the Service Request Enable register).

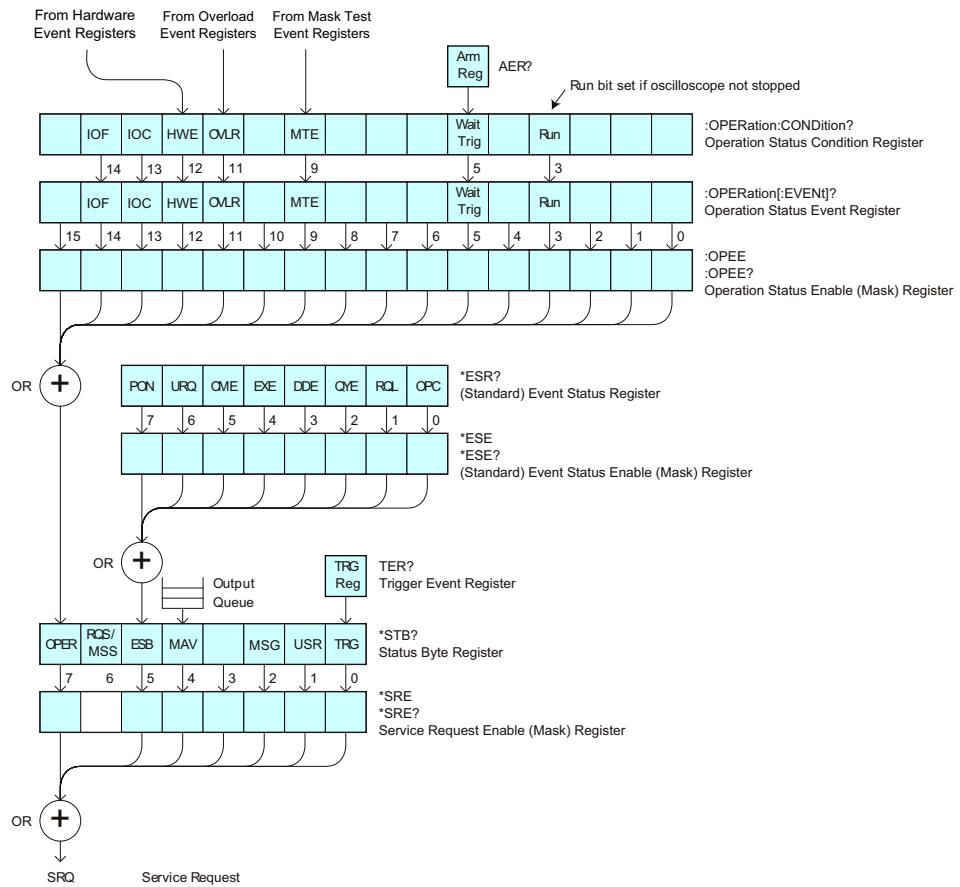
The Status Byte Register, the Standard Event Status Register group, and the Output Queue are defined as the Standard Status Data Structure Model in IEEE 488.2-1987.

The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled with the corresponding event enable register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. The \*CLS command clears all event registers and all queues except the output queue. If you send \*CLS immediately after a program message terminator, the output queue is also cleared.

## Status Reporting Data Structures

The following figure shows how the status register bits are masked and logically OR'ed to generate service requests (SRQ) on particular events.





The status register bits are described in more detail in the following tables:

- [Table 76](#)
- [Table 74](#)
- [Table 84](#)
- [Table 85](#)
- [Table 87](#)
- [Table 79](#)
- [Table 80](#)
- [Table 82](#)

The status registers picture above shows how the different status reporting data structures work together. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the \*ESE common command to set the corresponding bit in the Standard Event Status Enable Register.

To generate a service request (SRQ) interrupt to an external controller, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the \*SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

## Status Byte Register (STB)

The Status Byte Register is the summary-level register in the status reporting structure. It contains summary bits that monitor activity in the other status registers and queues. The Status Byte Register is a live register. That is, its summary bits are set and cleared by the presence and absence of a summary bit from other event registers or queues.

If the Status Byte Register is to be used with the Service Request Enable Register to set bit 6 (RQS/MSS) and to generate an SRQ, at least one of the summary bits must be enabled, then set. Also, event bits in all other status registers must be specifically enabled to generate the summary bit that sets the associated summary bit in the Status Byte Register.

The Status Byte Register can be read using either the \*STB? Common Command or the programming interface serial poll command. Both commands return the decimal-weighted sum of all set bits in the register. The difference between the two methods is that the serial poll command reads bit 6 as the Request Service (RQS) bit and clears the bit which clears the SRQ interrupt. The \*STB? command reads bit 6 as the Master Summary Status (MSS) and does not clear the bit or have any affect on the SRQ interrupt. The value returned is the total bit weights of all of the bits that are set at the present time.

The use of bit 6 can be confusing. This bit was defined to cover all possible computer interfaces, including a computer that could not do a serial poll. The important point to remember is that, if you are using an SRQ interrupt to an external computer, the serial poll command clears bit 6. Clearing bit 6 allows the oscilloscope to generate another SRQ interrupt when another enabled event occurs.

No other bits in the Status Byte Register are cleared by either the \*STB? query or the serial poll, except the Message Available bit (bit 4). If there are no other messages in the Output Queue, bit 4 (MAV) can be cleared as a result of reading the response to the \*STB? command.

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights. Since these bits were not enabled to generate an SRQ, bit 6 (weight = 64) is not set.

The following example uses the \*STB? query to read the contents of the oscilloscope's Status Byte Register.

```
myScope.WriteString "*STB?"
varQueryResult = myScope.ReadNumber
MsgBox "Status Byte Register, Read: 0x" + Hex(varQueryResult)
```

The next program prints 0xD1 and clears bit 6 (RQS) and bit 4 (MAV) of the Status Byte Register. The difference in the output value between this example and the previous one is the value of bit 6 (weight = 64). Bit 6 is set when the first enabled summary bit is set and is cleared when the Status Byte Register is read by the serial poll command.

- Example** The following example uses the resource session object's ReadSTB method to read the contents of the oscilloscope's Status Byte Register.

```
varQueryResult = myScope.IO.ReadSTB  
MsgBox "Status Byte Register, Serial Poll: 0x" + Hex(varQueryResult)
```

**NOTE**

**Use Serial Polling to Read Status Byte Register.** Serial polling is the preferred method to read the contents of the Status Byte Register because it resets bit 6 and allows the next enabled event that occurs to generate a new SRQ interrupt.

---

## Service Request Enable Register (SRE)

Setting the Service Request Enable Register bits enable corresponding bits in the Status Byte Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

Bits are set in the Service Request Enable Register using the \*SRE command and the bits that are set are read with the \*SRE? query.

**Example** The following example sets bit 4 (MAV) and bit 5 (ESB) in the Service Request Enable Register.

```
myScope.WriteString "*SRE " + CStr(CInt("&H30"))
```

This example uses the decimal parameter value of 48, the string returned by CStr(CInt("&H30")), to enable the oscilloscope to generate an SRQ interrupt under the following conditions:

- When one or more bytes in the Output Queue set bit 4 (MAV).
- When an enabled event in the Standard Event Status Register generates a summary bit that sets bit 5 (ESB).

## Trigger Event Register (TER)

This register sets the TRG bit in the status byte when a trigger event occurs.

The TER event register stays set until it is cleared by reading the register or using the \*CLS command. If your application needs to detect multiple triggers, the TER event register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation, you must clear the event register each time the trigger bit is set.

## Output Queue

The output queue stores the oscilloscope-to-controller responses that are generated by certain instrument commands and queries. The output queue generates the Message Available summary bit when the output queue contains one or more bytes. This summary bit sets the MAV bit (bit 4) in the Status Byte Register.

When using the Keysight VISA COM library, the output queue may be read with the FormattedIO488 object's ReadString, ReadNumber, ReadList, or ReadIEEEBlock methods.

## Message Queue

The message queue contains the text of the last message written to the advisory line on the screen of the oscilloscope. The length of the oscilloscope's message queue is 1. Note that messages sent with the :SYSTem:DSP command do not set the MSG status bit in the Status Byte Register.

## (Standard) Event Status Register (ESR)

The (Standard) Event Status Register (ESR) monitors the following oscilloscope status events:

- PON - Power On
- URQ - User Request
- CME - Command Error
- EXE - Execution Error
- DDE - Device Dependent Error
- QYE - Query Error
- RQC - Request Control
- OPC - Operation Complete

When one of these events occur, the event sets the corresponding bit in the register. If the bits are enabled in the Standard Event Status Enable Register, the bits set in this register generate a summary bit to set bit 5 (ESB) in the Status Byte Register.

You can read the contents of the Standard Event Status Register and clear the register by sending the \*ESR? query. The value returned is the total bit weights of all of the bits that are set at the present time.

**Example** The following example uses the \*ESR query to read the contents of the Standard Event Status Register.

```
myScope.WriteString "*ESR?"  
varQueryResult = myScope.ReadNumber  
MsgBox "Standard Event Status Register: 0x" + Hex(varQueryResult)
```

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights.

## (Standard) Event Status Enable Register (ESE)

To allow any of the (Standard) Event Status Register (ESR) bits to generate a summary bit, you must first enable that bit. Enable the bit by using the \*ESE (Event Status Enable) common command to set the corresponding bit in the (Standard) Event Status Enable Register (ESE).

Set bits are read with the \*ESE? query.

- Example** Suppose your application requires an interrupt whenever any type of error occurs. The error related bits in the (Standard) Event Status Register are bits 2 through 5 (hexadecimal value 0x3C). Therefore, you can enable any of these bits to generate the summary bit by sending:

```
myScope.WriteString "*ESE " + CStr(CInt("&H3C"))
```

Whenever an error occurs, it sets one of these bits in the (Standard) Event Status Register. Because all the error related bits are enabled, a summary bit is generated to set bit 5 (ESB) in the Status Byte Register.

If bit 5 (ESB) in the Status Byte Register is enabled (via the \*SRE command), an SRQ service request interrupt is sent to the controller PC.

### NOTE

**Disabled (Standard) Event Status Register bits respond but do not generate a summary bit.** (Standard) Event Status Register bits that are not enabled still respond to their corresponding conditions (that is, they are set if the corresponding event occurs). However, because they are not enabled, they do not generate a summary bit to the Status Byte Register.

## Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error 350, Queue overflow. Any time the queue overflows, the least recent errors remain in the queue, and the most recent error is discarded. The length of the oscilloscope's error queue is 30 (29 positions for the error messages, and 1 position for the Queue overflow message).

The error queue is read with the :SYSTem:ERRor? query. Executing this query reads and removes the oldest error from the head of the queue, which opens a position at the tail of the queue for a new error. When all the errors have been read from the queue, subsequent error queries return "0, No error".

The error queue is cleared when:

- the instrument is powered up,
- the instrument receives the \*CLS common command, or
- the last item is read from the error queue.

## Operation Status Event Register (:OPERegister[:EVENT])

The Operation Status Event Register register hosts these bits:

Name	Location	Description
RUN bit	bit 3	Is set whenever the instrument goes from a stop state to a single or running state.
WAIT TRIG bit	bit 5	Is set by the Trigger Armed Event Register and indicates that the trigger is armed.
MTE bit	bit 9	Comes from the Mask Test Event Registers.
OVLR bit	bit 11	Is set whenever a 50Ω input overload occurs.
HWE bit	bit 12	Comes from the Hardware Event Registers.
IOC bit	bit 13	Is set when the IO operation completes.
IOF bit	bit 14	Is set when the IO operation fails.

If any of these bits are set, the OPER bit (bit 7) of the Status Byte Register is set. The Operation Status Event Register is read and cleared with the :OPERegister[:EVENT]? query. The register output is enabled or disabled using the mask value supplied with the OPEE command.

## Operation Status Condition Register (:OPERegister:CONDition)

The Operation Status Condition Register register hosts these bits:

Name	Location	Description
RUN bit	bit 3	Is set whenever the instrument is not stopped.
WAIT TRIG bit	bit 5	Is set by the Trigger Armed Event Register and indicates that the trigger is armed.
MTE bit	bit 9	Comes from the Mask Test Event Registers.
OVLR bit	bit 11	Is set whenever a 50Ω input overload occurs.
HWE bit	bit 12	Comes from the Hardware Event Registers.
IOC bit	bit 13	Is set when the IO operation completes.
IOF bit	bit 14	Is set when the IO operation fails.

The :OPERegister:CONDition? query returns the value of the Operation Status Condition Register.

## Arm Event Register (AER)

This register sets bit 5 (Wait Trig bit) in the Operation Status Register and the OPER bit (bit 7) in the Status Byte Register when the instrument becomes armed.

The ARM event register stays set until it is cleared by reading the register with the AER? query or using the \*CLS command. If your application needs to detect multiple triggers, the ARM event register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation when the trigger bit is set, then you must clear the event register after each time it has been set.

## Overload Event Register (:OVLRegister)

The Overload Event Register register hosts these bits:

Name	Location	Description
Channel 1 OVL	bit 0	Overload has occurred on Channel 1 input.
Channel 2 OVL	bit 1	Overload has occurred on Channel 2 input.
Channel 3 OVL	bit 2	Overload has occurred on Channel 3 input.
Channel 4 OVL	bit 3	Overload has occurred on Channel 4 input.
External Trigger OVL	bit 4	Overload has occurred on External Trigger input.
Channel 1 Fault	bit 6	Fault has occurred on Channel 1 input.
Channel 2 Fault	bit 7	Fault has occurred on Channel 2 input.
Channel 3 Fault	bit 8	Fault has occurred on Channel 3 input.
Channel 4 Fault	bit 9	Fault has occurred on Channel 4 input.
External Trigger Fault	bit 10	Fault has occurred on External Trigger input.

## Hardware Event Event Register (:HWERegister[:EVENT])

This register hosts the PLL LOCKED bit (bit 12).

- The PLL LOCKED bit (bit 12) is for internal use and is not intended for general use.

## Hardware Event Condition Register (:HWERegister:CONDition)

This register hosts the PLL LOCKED bit (bit 12).

- The :HWERegister:CONDition? query returns the value of the Hardware Event Condition Register.
- The PLL LOCKED bit (bit 12) is for internal use and is not intended for general use.

## Mask Test Event Event Register (:MTERegister[:EVENT])

The Mask Test Event Event Register register hosts these bits:

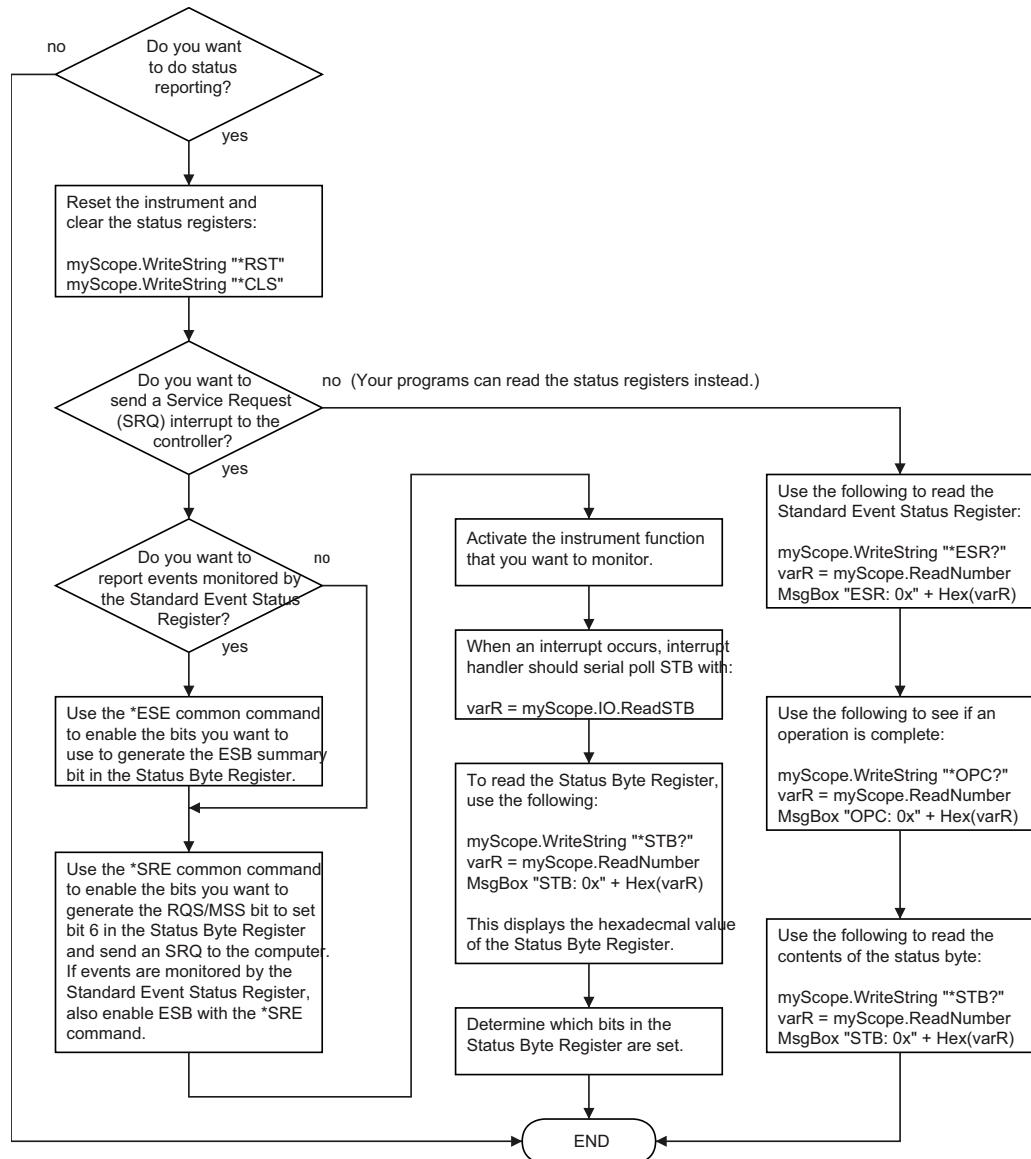
Name	Location	Description
Complete	bit 0	Is set when the mask test is complete.
Fail	bit 1	Is set when there is a mask test failure.
Started	bit 8	Is set when mask testing is started.
Auto Mask	bit 10	Is set when auto mask creation is completed.

The :MTERegister[:EVENT]? query returns the value of, and clears, the Mask Test Event Event Register.

## Clearing Registers and Queues

The \*CLS common command clears all event registers and all queues except the output queue. If \*CLS is sent immediately after a program message terminator, the output queue is also cleared.

## Status Reporting Decision Chart





# 40 Synchronizing Acquisitions

Synchronization in the Programming Flow /	1324
Blocking Synchronization /	1325
Polling Synchronization With Timeout /	1326
Synchronizing with a Single-Shot Device Under Test (DUT) /	1328
Synchronization with an Averaging Acquisition /	1330

When remotely controlling an oscilloscope with programming commands, it is often necessary to know when the oscilloscope has finished the previous operation and is ready for the next command. The most common example is when an acquisition is started using the :DIGitize, :RUN, or :SINGle commands. Before a measurement result can be queried, the acquisition must complete. Too often fixed delays are used to accomplish this wait, but fixed delays often use excessive time or the time may not be long enough. A better solution is to use synchronous commands and status to know when the oscilloscope is ready for the next request.

## Synchronization in the Programming Flow

Most remote programming follows these three general steps:

- 1 Set up the oscilloscope and device under test (see [page 1324](#)).
- 2 Acquire a waveform (see [page 1324](#)).
- 3 Retrieve results (see [page 1324](#)).

### Set Up the Oscilloscope

Before making changes to the oscilloscope setup, it is best to make sure it is stopped using the :STOP command followed by the \*OPC? query.

#### NOTE

It is not necessary to use \*OPC?, hard coded waits, or status checking when setting up the oscilloscope. After the oscilloscope is configured, it is ready for an acquisition.

### Acquire a Waveform

When acquiring a waveform there are two possible methods used to wait for the acquisition to complete. These methods are blocking and polling. The table below details when each method should be chosen and why.

	Blocking Wait	Polling Wait
Use When	You know the oscilloscope <i>will</i> trigger based on the oscilloscope setup and device under test.	You know the oscilloscope <i>may or may not</i> trigger on the oscilloscope setup and device under test.
Advantages	No need for polling. Fastest method.	Remote interface will not timeout No need for device clear if no trigger.
Disadvantages	Remote interface may timeout. Device clear only way to get control of oscilloscope if there is no trigger.	Slower method. Requires polling loop. Requires known maximum wait time.
Implementation Details	See " <a href="#">Blocking Synchronization</a> " on page 1325.	See " <a href="#">Polling Synchronization With Timeout</a> " on page 1326.

### Retrieve Results

Once the acquisition is complete, it is safe to retrieve measurements and statistics.

## Blocking Synchronization

Use the :DIGitize command to start the acquisition. This blocks subsequent queries until the acquisition and processing is complete. For example:

```

'
' Synchronizing acquisition using blocking.
' =====

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Sub Main()

    On Error GoTo VisaComError

    ' Create the VISA COM I/O resource.
    Set myMgr = New VisaComLib.ResourceManager
    Set myScope = New VisaComLib.FormattedIO488
    Set myScope.IO = myMgr.Open("TCPIP0::130.29.69.12::inst0::INSTR")
    myScope.IO.Clear      ' Clear the interface.

    ' Set up.
    ' -----
    myScope.WriteString ":TRIGger:MODE EDGE"
    myScope.WriteString ":TRIGger:EDGE:LEVel 2"
    myScope.WriteString ":TIMEbase:SCALe 5e-8"

    ' Acquire.
    ' -----
    myScope.WriteString ":DIGitize"

    ' Get results.
    ' -----
    myScope.WriteString ":MEASure:RISetime"
    myScope.WriteString ":MEASure:RISetime?"
    varQueryResult = myScope.ReadNumber      ' Read risetime.
    Debug.Print "Risetime: " + _
               FormatNumber(varQueryResult * 1000000000, 1) + " ns"

    Exit Sub

VisaComError:
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

## Polling Synchronization With Timeout

This example requires a timeout value so the operation can abort if an acquisition does not occur within the timeout period:

```

'
' Synchronizing acquisition using polling.
' =====

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub Main()

On Error GoTo VisaComError

' Create the VISA COM I/O resource.
Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488
Set myScope.IO = myMgr.Open("TCPIP0::130.29.69.12::inst0::INSTR")
myScope.IO.Clear      ' Clear the interface.

' Set up.
' -----
' Set up the trigger and horizontal scale.
myScope.WriteString ":TRIGger:MODE EDGE"
myScope.WriteString ":TRIGger:EDGE:LEVel 2"
myScope.WriteString ":TIMEbase:SCALE 5e-8"

' Stop acquisitions and wait for the operation to complete.
myScope.WriteString ":STOP"
myScope.WriteString "*OPC?"
strQueryResult = myScope.ReadString

' Acquire.
' -----
' Start a single acquisition.
myScope.WriteString ":SINGLE"

' Oscilloscope is armed and ready, enable DUT here.
Debug.Print "Oscilloscope is armed and ready, enable DUT."

' Look for RUN bit = stopped (acquisition complete).
Dim lngTimeout As Long      ' Max millisecs to wait for single-shot.
Dim lngElapsed As Long
lngTimeout = 10000      ' 10 seconds.
lngElapsed = 0

Do While lngElapsed <= lngTimeout

```

```
myScope.WriteString ":OPERegister:CONDITION?"
varQueryResult = myScope.ReadNumber
' Mask RUN bit (bit 3, &H8).
If (varQueryResult And &H8) = 0 Then
    Exit Do
Else
    Sleep 100      ' Small wait to prevent excessive queries.
    lngElapsed = lngElapsed + 100
End If
Loop

' Get results.
' -----
If lngElapsed < lngTimeout Then
    myScope.WriteString ":MEASure:RISetime"
    myScope.WriteString ":MEASure:RISetime?"
    varQueryResult = myScope.ReadNumber      ' Read risetime.
    Debug.Print "Risetime: " + _
        FormatNumber(varQueryResult * 1000000000, 1) + " ns"
Else
    Debug.Print "Timeout waiting for single-shot trigger."
End If

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub
```

## Synchronizing with a Single-Shot Device Under Test (DUT)

The examples in "Blocking Synchronization" on page 1325 and "Polling Synchronization With Timeout" on page 1326 assume the DUT is continually running and therefore the oscilloscope will have more than one opportunity to trigger. With a single shot DUT, there is only one opportunity for the oscilloscope to trigger, so it is necessary for the oscilloscope to be armed and ready before the DUT is enabled.

### NOTE

The blocking :DIGitize command cannot be used for a single shot DUT because once the :DIGITIZE command is issued, the oscilloscope is blocked from any further commands until the acquisition is complete.

This example is the same "Polling Synchronization With Timeout" on page 1326 with the addition of checking for the armed event status.

```
' Synchronizing single-shot acquisition using polling.
' =====

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub Main()

    On Error GoTo VisaComError

    ' Create the VISA COM I/O resource.
    Set myMgr = New VisaComLib.ResourceManager
    Set myScope = New VisaComLib.FormattedIO488
    Set myScope.IO = myMgr.Open("TCPIP0::130.29.69.12::inst0::INSTR")
    myScope.IO.Clear      ' Clear the interface.

    ' Set up.
    ' -----
    ' Set up the trigger and horizontal scale.
    myScope.WriteString ":TRIGGER:MODE EDGE"
    myScope.WriteString ":TRIGGER:EDGE:LEVel 2"
    myScope.WriteString ":TIMEbase:SCALE 5e-8"

    ' Stop acquisitions and wait for the operation to complete.
    myScope.WriteString ":STOP"
    myScope.WriteString "*OPC?"
    strQueryResult = myScope.ReadString

    ' Acquire.
```

```

' -----
' Start a single acquisition.
myScope.WriteString ":SINGLE"

' Wait until the trigger system is armed.
Do
    Sleep 100      ' Small wait to prevent excessive queries.
    myScope.WriteString ":AER?"
    varQueryResult = myScope.ReadNumber
Loop Until varQueryResult = 1

' Oscilloscope is armed and ready, enable DUT here.
Debug.Print "Oscilloscope is armed and ready, enable DUT."

' Now, look for RUN bit = stopped (acquisition complete).
Dim lngTimeout As Long      ' Max millisecs to wait for single-shot.
Dim lngElapsed As Long
lngTimeout = 10000      ' 10 seconds.
lngElapsed = 0

Do While lngElapsed <= lngTimeout
    myScope.WriteString ":OPERRegister:CONDITION?"
    varQueryResult = myScope.ReadNumber
    ' Mask RUN bit (bit 3, &H8).
    If (varQueryResult And &H8) = 0 Then
        Exit Do
    Else
        Sleep 100      ' Small wait to prevent excessive queries.
        lngElapsed = lngElapsed + 100
    End If
Loop

' Get results.
' -----
If lngElapsed < lngTimeout Then
    myScope.WriteString ":MEASure:RISetime"
    myScope.WriteString ":MEASure:RISetime?"
    varQueryResult = myScope.ReadNumber      ' Read risetime.
    Debug.Print "Risetime: " + _
        FormatNumber(varQueryResult * 1000000000, 1) + " ns"
Else
    Debug.Print "Timeout waiting for single-shot trigger."
End If

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

## Synchronization with an Averaging Acquisition

When averaging, it is necessary to know when the average count has been reached. The :SINGle command does not average.

If it is known that a trigger will occur, a :DIGItize will acquire the complete number of averages, but if the number of averages is large, a timeout on the connection can occur.

The example below polls during the :DIGItize to prevent a timeout on the connection.

```
' Synchronizing in averaging acquisition mode.
' =====

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Sub Main()

    On Error GoTo VisaComError

    ' Create the VISA COM I/O resource.
    Set myMgr = New VisaComLib.ResourceManager
    Set myScope = New VisaComLib.FormattedIO488
    Set myScope.IO = myMgr.Open("TCPIP0::130.29.69.12::inst0::INSTR")
    myScope.IO.Clear      ' Clear the interface.
    myScope.IO.Timeout = 5000

    ' Set up.
    ' -----
    ' Set up the trigger and horizontal scale.
    myScope.WriteString ":TRIGger:SWEep NORMal"
    myScope.WriteString ":TRIGger:MODE EDGE"
    myScope.WriteString ":TRIGger:EDGE:LEVel 2"
    myScope.WriteString ":TIMEbase:SCALe 5e-8"

    ' Stop acquisitions and wait for the operation to complete.
    myScope.WriteString ":STOP"
    myScope.WriteString "*OPC?"
    strQueryResult = myScope.ReadString

    ' Set up average acquisition mode.
    Dim lngAverages As Long
    lngAverages = 256
    myScope.WriteString ":ACQuire:COUNT " + CStr(lngAverages)
    myScope.WriteString ":ACQuire:TYPE AVERAGE"
```

```

' Save *ESE (Standard Event Status Enable register) mask
' (so it can be restored later).
Dim varInitialESE As Variant
myScope.WriteString "*ESE?"
varInitialESE = myScope.ReadNumber

' Set *ESE mask to allow only OPC (Operation Complete) bit.
myScope.WriteString "*ESE " + CStr(CInt("&H01"))

' Acquire using :DIGitize. Set up OPC bit to be set when the
' operation is complete.
' -----
myScope.WriteString ":DIGitize"
myScope.WriteString "*OPC"

' Assume the oscilloscope will trigger, if not put a check here.

' Wait until OPC becomes true (bit 5 of Status Byte register, STB,
' from Standard Event Status register, ESR, is set). STB can be
' read during :DIGitize without generating a timeout.
Do
    Sleep 4000      ' Poll more often than the timeout setting.
    varQueryResult = myScope.IO.ReadSTB
Loop While (varQueryResult And &H20) = 0

' Clear ESR and restore previously saved *ESE mask.
myScope.WriteString "*ESR?"      ' Clear ESR by reading it.
varQueryResult = myScope.ReadNumber
myScope.WriteString "*ESE " + CStr(varInitialESE)

' Get results.
' -----
myScope.WriteString ":WAVeform:COUNt?"
varQueryResult = myScope.ReadNumber
Debug.Print "Averaged waveforms: " + CStr(varQueryResult)

myScope.WriteString ":MEASure:RISetime"
myScope.WriteString ":MEASure:RISetime?"
varQueryResult = myScope.ReadNumber      ' Read risetime.
Debug.Print "Risetime: " +
    FormatNumber(varQueryResult * 1000000000, 1) + " ns"

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```



# 41 More About Oscilloscope Commands

- Command Classifications / 1334
- Valid Command/Query Strings / 1335
- Query Return Values / 1341
- All Oscilloscope Commands Are Sequential / 1342

## Command Classifications

To help you use existing programs with your oscilloscope, or use current programs with the next generation of Keysight InfiniiVision oscilloscopes, commands are classified by the following categories:

- ["Core Commands"](#) on page 1334
- ["Non-Core Commands"](#) on page 1334
- ["Obsolete Commands"](#) on page 1334

### **C** Core Commands

Core commands are a common set of commands that provide basic oscilloscope functionality on this oscilloscope and future Keysight InfiniiVision oscilloscopes. Core commands are unlikely to be modified in the future. If you restrict your programs to core commands, the programs should work across product offerings in the future, assuming appropriate programming methods are employed.

### **N** Non-Core Commands

Non-core commands are commands that provide specific features, but are not universal across all Keysight InfiniiVision oscilloscope models. Non-core commands may be modified or deleted in the future. With a command structure as complex as the one for your oscilloscope, some evolution over time is inevitable. Keysight's intent is to continue to expand command subsystems, such as the rich and evolving trigger feature set.

### **O** Obsolete Commands

Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs. Generally, these commands are mapped onto some of the Core and Non-core commands, but may not strictly have the same behavior as the new command. None of the obsolete commands are guaranteed to remain functional in future products. New systems and programs should use the Core (and Non-core) commands. Obsolete commands are listed in:

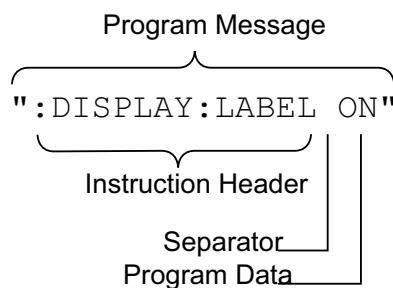
- [Chapter 37](#), “Obsolete and Discontinued Commands,” starting on page 1233

## Valid Command/Query Strings

- ["Program Message Syntax" on page 1335](#)
- ["Duplicate Mnemonics" on page 1339](#)
- ["Tree Traversal Rules and Multiple Commands" on page 1339](#)

### Program Message Syntax

To program the instrument remotely, you must understand the command format and structure expected by the instrument. The IEEE 488.2 syntax rules govern how individual elements such as headers, separators, program data, and terminators may be grouped together to form complete instructions. Syntax definitions are also given to show how query responses are formatted. The following figure shows the main syntactical parts of a typical program statement.



Instructions (both commands and queries) normally appear as a string embedded in a statement of your host language, such as Visual Basic or C/C++. The only time a parameter is not meant to be expressed as a string is when the instruction's syntax definition specifies <block data>, such as <learn string>. There are only a few instructions that use block data.

Program messages can have long or short form commands (and data in some cases – see ["Long Form to Short Form Truncation Rules" on page 1336](#)), and upper and/or lower case ASCII characters may be used. (Query responses, however, are always returned in upper case.)

Instructions are composed of two main parts:

- The header, which specifies the command or query to be sent.
- The program data, which provide additional information needed to clarify the meaning of the instruction.

**Instruction Header** The instruction header is one or more mnemonics separated by colons (:) that represent the operation to be performed by the instrument.

"":DISPLAY:LABEL ON" is a command. Queries are indicated by adding a question mark (?) to the end of the header, for example, "":DISPLAY:LABEL?". Many instructions can be used as either commands or queries, depending on whether or

not you have included the question mark. The command and query forms of an instruction usually have different program data. Many queries do not use any program data.

There are three types of headers:

- ["Simple Command Headers" on page 1337](#)
- ["Compound Command Headers" on page 1337](#)
- ["Common Command Headers" on page 1337](#)

**White Space (Separator)** White space is used to separate the instruction header from the program data. If the instruction does not require any program data parameters, you do not need to include any white space. White space is defined as one or more space characters. ASCII defines a space to be character 32 (in decimal).

**Program Data** Program data are used to clarify the meaning of the command or query. They provide necessary information, such as whether a function should be on or off, or which waveform is to be displayed. Each instruction's syntax definition shows the program data, as well as the values they accept. ["Program Data Syntax Rules" on page 1338](#) describes all of the general rules about acceptable values.

When there is more than one data parameter, they are separated by commas(,). Spaces can be added around the commas to improve readability.

**Program Message Terminator** The program instructions within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Or-Identify) asserted in the programming interface, or a combination of the two. Asserting the EOI sets the EOI control line low on the last byte of the data message. The NL character is an ASCII linefeed (decimal 10).

#### NOTE

**New Line Terminator Functions.** The NL (New Line) terminator has the same function as an EOS (End Of String) and EOT (End Of Text) terminator.

### Long Form to Short Form Truncation Rules

To get the short form of a command/keyword:

- When the command/keyword is longer than four characters, use the first four characters of the command/keyword unless the fourth character is a vowel; when the fourth character is a vowel, use the first three characters of the command/keyword.
- When the command/keyword is four or fewer characters, use all of the characters.

Long Form	Short form
RANGE	RANG
PATTERn	PATT

Long Form	Short form
TIMebase	TIM
DELay	DEL
TYPE	TYPE

In the oscilloscope programmer's documentation, the short form of a command is indicated by uppercase characters.

Programs written in long form are easily read and are almost self-documenting. The short form syntax conserves the amount of controller memory needed for program storage and reduces I/O activity.

### Simple Command Headers

Simple command headers contain a single mnemonic. :AUToscale and :DIGitize are examples of simple command headers typically used in the oscilloscope. The syntax is:

```
<program mnemonic><terminator>
```

Simple command headers must occur at the beginning of a program message; if not, they must be preceded by a colon.

When program data must be included with the simple command header (for example, :DIGITIZE CHANnel1), white space is added to separate the data from the header. The syntax is:

```
<program mnemonic><separator><program data><terminator>
```

### Compound Command Headers

Compound command headers are a combination of two or more program mnemonics. The first mnemonic selects the subsystem, and the second mnemonic selects the function within that subsystem. The mnemonics within the compound message are separated by colons. For example, to execute a single function within a subsystem:

```
:<subsystem>:<function><separator><program data><terminator>
```

For example, :CHANnel1:BWLIMIT ON

### Common Command Headers

Common command headers control IEEE 488.2 functions within the instrument (such as clear status). Their syntax is:

```
*<command header><terminator>
```

No space or separator is allowed between the asterisk (\*) and the command header. \*CLS is an example of a common command header.

## Program Data Syntax Rules

Program data is used to convey a parameter information related to the command header. At least one space must separate the command header or query header from the program data.

```
<program mnemonic><separator><data><terminator>
```

When a program mnemonic or query has multiple program data, a comma separates sequential program data.

```
<program mnemonic><separator><data>,<data><terminator>
```

For example, :MEASure:DELay CHANnel1,CHANnel2 has two program data: CHANnel1 and CHANnel2.

Two main types of program data are used in commands: character and numeric.

### Character Program Data

Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the :TIMEbase:MODE command can be set to normal, zoomed (delayed), XY, or ROLL. The character program data in this case may be MAIN, WINDOW, XY, or ROLL. The command :TIMEbase:MODE WINDOW sets the time base mode to zoomed.

The available mnemonics for character program data are always included with the command's syntax definition.

When sending commands, you may either the long form or short form (if one exists). Uppercase and lowercase letters may be mixed freely.

When receiving query responses, uppercase letters are used exclusively.

### Numeric Program Data

Some command headers require program data to be expressed numerically. For example, :TIMEbase:RANGE requires the desired full scale range to be expressed numerically.

For numeric program data, you have the option of using exponential notation or using suffix multipliers to indicate the numeric value. The following numbers are all equal:

```
28 = 0.28E2 = 280e-1 = 28000m = 0.028K = 28e-3K.
```

When a syntax definition specifies that a number is an integer, that means that the number should be whole. Any fractional part will be ignored, truncating the number. Numeric data parameters accept fractional values are called real numbers.

All numbers must be strings of ASCII characters. Thus, when sending the number 9, you would send a byte representing the ASCII code for the character 9 (which is 57). A three-digit number like 102 would take up three bytes (ASCII codes 49, 48, and 50). This is handled automatically when you include the entire instruction in a string.

## Duplicate Mnemonics

Identical function mnemonics can be used in more than one subsystem. For example, the function mnemonic RANGe may be used to change the vertical range or to change the horizontal range:

```
:CHANnel1:RANGE .4
```

Sets the vertical range of channel 1 to 0.4 volts full scale.

```
:TIMEbase:RANGE 1
```

Sets the horizontal time base to 1 second full scale.

:CHANnel1 and :TIMEbase are subsystem selectors and determine which range is being modified.

## Tree Traversal Rules and Multiple Commands

Command headers are created by traversing down the command tree. A legal command header would be :TIMEbase:RANGe. This is referred to as a *compound header*. A compound header is a header made of two or more mnemonics separated by colons. The mnemonic created contains no spaces.

The following rules apply to traversing the tree:

- A leading colon (<NL> or EOI true on the last byte) places the parser at the root of the command tree. A leading colon is a colon that is the first character of a program header. Executing a subsystem command lets you access that subsystem until a leading colon or a program message terminator (<NL>) or EOI true is found.
- In the command tree, use the last mnemonic in the compound header as the reference point (for example, RANGe). Then find the last colon above that mnemonic (TIMEbase:). That is the point where the parser resides. Any command below that point can be sent within the current program message without sending the mnemonics which appear above them (for example, POSition).

The output statements in the examples are written using the Keysight VISA COM library in Visual Basic. The quoted string is placed on the bus, followed by a carriage return and linefeed (CRLF).

To execute more than one function within the same subsystem, separate the functions with a semicolon (;):

```
:<subsystem>:<function><separator><data>;<function><separator><data><terminator>
```

For example:

```
myScope.WriteString ":TIMEbase:RANGE 0.5;POSITION 0"
```

**NOTE**

The colon between TIMebase and RANGe is necessary because TIMebase:RANGE is a compound command. The semicolon between the RANGe command and the POSition command is the required program message unit separator. The POSition command does not need TIMebase preceding it because the TIMebase:RANGE command sets the parser to the TIMebase node in the tree.

**Example 2:**  
**Program Message Terminator Sets Parser Back to Root**

```
myScope.WriteString ":TIMebase:REFerence CENTER;POSITION 0.00001"
or
myScope.WriteString ":TIMebase:REFerence CENTER"
myScope.WriteString ":TIMebase:POSITION 0.00001"
```

**NOTE**

In the first line of example 2, the subsystem selector is implied for the POSITION command in the compound command. The POSITION command must be in the same program message as the REFerence command because the program message terminator places the parser back at the root of the command tree.

A second way to send these commands is by placing TIMebase: before the POSITION command as shown in the second part of example 2. The space after POSITION is required.

**Example 3:**  
**Selecting Multiple Subsystems**

You can send multiple program commands and program queries for different subsystems on the same line by separating each command with a semicolon. The colon following the semicolon enables you to enter a new subsystem. For example:

```
<program mnemonic><data>;<program mnemonic><data><terminator>
```

For example:

```
myScope.WriteString ":TIMebase:REFerence CENTER;:DISPlay:VECTors ON"
```

**NOTE**

The leading colon before DISPlay:VECTors ON tells the parser to go back to the root of the command tree. The parser can then see the DISPlay:VECTors ON command. The space between REFerence and CENter is required; so is the space between VECTors and ON.

Multiple commands may be any combination of compound and simple commands.

## Query Return Values

Command headers immediately followed by a question mark (?) are queries. Queries are used to get results of measurements made by the instrument or to find out how the instrument is currently configured.

After receiving a query, the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued.

When read, the answer is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMEbase:RANGE? places the current time base setting in the output queue. When using the Keysight VISA COM library in Visual Basic, the controller statements:

```
Dim strQueryResult As String  
myScope.WriteString ":TIMEbase:RANGE?"  
strQueryResult = myScope.ReadString
```

pass the value across the bus to the controller and place it in the variable strQueryResult.

### NOTE

**Read Query Results Before Sending Another Command.** Sending another command or query before reading the result of a query clears the output buffer (the current response) and places a Query INTERRUPTED error in the error queue.

### Infinity Representation

The representation of infinity is +9.9E+37. This is also the value returned when a measurement cannot be made.

## All Oscilloscope Commands Are Sequential

IEEE 488.2 makes the distinction between sequential and overlapped commands:

- *Sequential commands* finish their task before the execution of the next command starts.
- *Overlapped commands* run concurrently. Commands following an overlapped command may be started before the overlapped command is completed.

All of the oscilloscope commands are sequential.

## 42 Programming Examples

VISA COM Examples / 1344

VISA Examples / 1377

SICL Examples / 1430

SCPI.NET Examples / 1450

Example programs are ASCII text files that can be cut from the help file and pasted into your favorite text editor.

## VISA COM Examples

- "VISA COM Example in Visual Basic" on page 1344
- "VISA COM Example in C#" on page 1353
- "VISA COM Example in Visual Basic .NET" on page 1362
- "VISA COM Example in Python" on page 1370

### VISA COM Example in Visual Basic

To run this example in Visual Basic for Applications (VBA):

- 1 Start the application that provides Visual Basic for Applications (for example, Microsoft Excel).
- 2 Press ALT+F11 to launch the Visual Basic editor.
- 3 Reference the Keysight VISA COM library:
  - a Choose **Tools>References...** from the main menu.
  - b In the References dialog, check the "VISA COM 5.5 Type Library".
  - c Click **OK**.
- 4 Choose **Insert>Module**.
- 5 Cut-and-paste the code that follows into the editor.
- 6 Edit the program to use the VISA address of your oscilloscope, and save the changes.
- 7 Run the program.

```

'
' Keysight VISA COM Example in Visual Basic
' -----
' This program illustrates a few commonly-used programming
' features of your Keysight oscilloscope.
' -----


Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

' For Sleep subroutine.
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'
' Main Program
' -----


Sub Main()

```

```

On Error GoTo VisaComError

' Create the VISA COM I/O resource.
Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488
Set myScope.IO =
    myMgr.Open("USB0::0x0957::0x17A6::US50210029::0::INSTR")
myScope.IO.Clear      ' Clear the interface.
myScope.IO.Timeout = 10000   ' Set I/O communication timeout.

' Initialize - start from a known state.
Initialize

' Capture data.
Capture

' Analyze the captured waveform.
Analyze

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description
End

End Sub

'
' Initialize the oscilloscope to a known state.
' -----
Private Sub Initialize()

On Error GoTo VisaComError

' Get and display the device's *IDN? string.
strQueryResult = DoQueryString("*IDN?")
Debug.Print "Identification string: " + strQueryResult

' Clear status and load the default setup.
DoCommand "*CLS"
DoCommand "*RST"

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description
End

End Sub

'
' Capture the waveform.
' -----
Private Sub Capture()

```

```

On Error GoTo VisaComError

' Use auto-scale to automatically configure oscilloscope.
' -----
DoCommand ":AUToscale"

' Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
DoCommand ":TRIGger:MODE EDGE"
Debug.Print "Trigger mode: " + _
    DoQueryString(":TRIGger:MODE?")

' Set EDGE trigger parameters.
DoCommand ":TRIGger:EDGE:SOURCe CHANnel1"
Debug.Print "Trigger edge source: " + _
    DoQueryString(":TRIGger:EDGE:SOURce?")

DoCommand ":TRIGger:EDGE:LEVel 1.5"
Debug.Print "Trigger edge level: " + _
    DoQueryString(":TRIGger:EDGE:LEVel?")

DoCommand ":TRIGger:EDGE:SLOPe POSitive"
Debug.Print "Trigger edge slope: " + _
    DoQueryString(":TRIGger:EDGE:SLOPe?")

' Save oscilloscope configuration.
' -----
varQueryResult = DoQueryIEEEBlock_UI1(":SYSTem:SETup?")

' Output setup string to a file:
Dim strPath As String
strPath = "c:\scope\config\setup.dat"
Dim hFile As Long
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Put hFile, , varQueryResult ' Write data.
Close hFile ' Close file.
Debug.Print "Setup bytes saved: " + CStr(LenB(varQueryResult))

' Change settings with individual commands:
' -----

' Set vertical scale and offset.
DoCommand ":CHANnel1:SCALe 0.05"
Debug.Print "Channel 1 vertical scale: " + _
    DoQueryString(":CHANnel1:SCALe?")

DoCommand ":CHANnel1:OFFSet -1.5"
Debug.Print "Channel 1 vertical offset: " + _
    DoQueryString(":CHANnel1:OFFSet?")

' Set horizontal scale and offset.
DoCommand ":TIMEbase:SCALe 0.0002"
Debug.Print "Timebase scale: " + _
    DoQueryString(":TIMEbase:SCALe?")

DoCommand ":TIMEbase:POSItion 0.0"
Debug.Print "Timebase position: " + _

```

```

DoQueryString(":TIMEbase:POSITION?")

' Set the acquisition type (NORMal, PEAK, AVERage, or HRESolution).
DoCommand ":ACQuire:TYPE NORMal"
Debug.Print "Acquire type: " + _
DoQueryString(":ACQuire:TYPE?")

' Or, configure by loading a previously saved setup.
' -----
Dim varSetupString As Variant
strPath = "c:\scope\config\setup.dat"
Open strPath For Binary Access Read As hFile    ' Open file for input.
Get hFile, , varSetupString    ' Read data.
Close hFile    ' Close file.
' Write learn string back to oscilloscope using ":SYSTem:SETup"
' command:
DoCommandIEEEBlock ":SYSTem:SETup", varSetupString
Debug.Print "Setup bytes restored: " + CStr(LenB(varSetupString))

' Capture an acquisition using :DIGItize.
' -----
DoCommand ":DIGItize CHANnel1"

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description
End

End Sub

'
' Analyze the captured waveform.
' -----
Private Sub Analyze()

On Error GoTo VisaComError

' Make a couple of measurements.
' -----
DoCommand ":MEASure:SOURce CHANnel1"
Debug.Print "Measure source: " + _
DoQueryString(":MEASure:SOURce?")

DoCommand ":MEASure:FREQuency"
varQueryResult = DoQueryNumber(":MEASure:FREQuency?")
MsgBox "Frequency:" + vbCrLf + _
FormatNumber(varQueryResult / 1000, 4) + " kHz"

DoCommand ":MEASure:VAMPplitude"
varQueryResult = DoQueryNumber(":MEASure:VAMPplitude?")
MsgBox "Vertical amplitude:" + vbCrLf + _
FormatNumber(varQueryResult, 4) + " V"

' Download the screen image.

```

```

' -----
' Get screen image.
DoCommand ":HARDcopy:INKSaver OFF"
Dim byteData() As Byte
byteData = DoQueryIEEEBlock_UI1(":DISPLAY:DATA? PNG, COLOR")

' Save screen image to a file.
Dim strPath As String
strPath = "c:\scope\data\screen.png"
If Len(Dir(strPath)) Then
    Kill strPath      ' Remove file if it exists.
End If

Dim hFile As Long
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Put hFile, , byteData      ' Write data.
Close hFile      ' Close file.
MsgBox "Screen image (" + CStr(UBound(byteData) + 1) + _
       " bytes) written to " + strPath

' Download waveform data.
' -----

' Set the waveform points mode.
DoCommand ":WAVeform:POINTs:MODE RAW"
Debug.Print "Waveform points mode: " + _
           DoQueryString(":WAVeform:POINTs:MODE?")

' Get the number of waveform points available.
Debug.Print "Waveform points available: " + _
           DoQueryString(":WAVeform:POINTs?")

' Set the waveform source.
DoCommand ":WAVeform:SOURce CHANnel1"
Debug.Print "Waveform source: " + _
           DoQueryString(":WAVeform:SOURce?")

' Choose the format of the data returned (WORD, BYTE, ASCII):
DoCommand ":WAVeform:FORMAT BYTE"
Debug.Print "Waveform format: " + _
           DoQueryString(":WAVeform:FORMAT?")

' Display the waveform settings:
Dim Preamble()
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim sngYOrigin As Single
Dim lngYReference As Long

```

```

Preamble() = DoQueryNumbers(":WAVeform:PREamble?")

intFormat = Preamble(0)
intType = Preamble(1)
lngPoints = Preamble(2)
lngCount = Preamble(3)
dblXIncrement = Preamble(4)
dblXOrigin = Preamble(5)
lngXReference = Preamble(6)
sngYIncrement = Preamble(7)
sngYOrigin = Preamble(8)
lngYReference = Preamble(9)

If intFormat = 0 Then
    Debug.Print "Waveform format: BYTE"
ElseIf intFormat = 1 Then
    Debug.Print "Waveform format: WORD"
ElseIf intFormat = 4 Then
    Debug.Print "Waveform format: ASCii"
End If

If intType = 0 Then
    Debug.Print "Acquisition type: NORMAL"
ElseIf intType = 1 Then
    Debug.Print "Acquisition type: PEAK"
ElseIf intType = 2 Then
    Debug.Print "Acquisition type: AVERage"
ElseIf intType = 3 Then
    Debug.Print "Acquisition type: HRESolution"
End If

Debug.Print "Waveform points: " + _
FormatNumber(lngPoints, 0)

Debug.Print "Waveform average count: " + _
FormatNumber(lngCount, 0)

Debug.Print "Waveform X increment: " + _
Format(dblXIncrement, "Scientific")

Debug.Print "Waveform X origin: " + _
Format(dblXOrigin, "Scientific")

Debug.Print "Waveform X reference: " + _
FormatNumber(lngXReference, 0)

Debug.Print "Waveform Y increment: " + _
Format(sngYIncrement, "Scientific")

Debug.Print "Waveform Y origin: " + _
FormatNumber(sngYOrigin, 0)

Debug.Print "Waveform Y reference: " + _
FormatNumber(lngYReference, 0)

' Get the waveform data
varQueryResult = DoQueryIEEEBlock_UI1(":WAVeform:DATA?")

```

```

Debug.Print "Number of data values: " + _
CStr(UBound(varQueryResult) + 1)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"

' Open file for output.
Open strPath For Output Access Write Lock Write As hFile

' Output waveform data in CSV format.
Dim lngDataValue As Long
Dim lngI As Long

For lngI = 0 To UBound(varQueryResult)
    lngDataValue = varQueryResult(lngI)

    ' Write time value, voltage value.
    Print #hFile, _
        FormatNumber(dblXOrigin + (lngI * dblXIncrement), 9) + _
        ", " + _
        FormatNumber(((lngDataValue - lngYReference) * _
        sngYIncrement) + sngYOrigin)

    Next lngI

    ' Close output file.
    Close hFile ' Close file.
    MsgBox "Waveform format BYTE data written to " + _
        "c:\scope\data\waveform_data.csv."

    Exit Sub

VisaComError:
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description
End

End Sub

Private Sub DoCommand(command As String)

    On Error GoTo VisaComError

    myScope.WriteString command
    CheckInstrumentErrors

    Exit Sub

VisaComError:
    MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
        Err.Source + ", " + _
        Err.Description, vbExclamation, "VISA COM Error"
End

End Sub

Private Sub DoCommandIEEEBlock(command As String, data As Variant)

```

```

On Error GoTo VisaComError

Dim strErrors As String

myScope.WriteLineEEBlock command, data
CheckInstrumentErrors

Exit Sub

VisaComError:
MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
       Err.Source + ", " + _
       Err.Description, vbExclamation, "VISA COM Error"
End

End Sub

Private Function DoQueryString(query As String) As String

On Error GoTo VisaComError

myScope.WriteString query
DoQueryString = myScope.ReadString
CheckInstrumentErrors

Exit Function

VisaComError:
MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
       Err.Source + ", " + _
       Err.Description, vbExclamation, "VISA COM Error"
End

End Function

Private Function DoQueryNumber(query As String) As Variant

On Error GoTo VisaComError

myScope.WriteString query
DoQueryNumber = myScope.ReadNumber
CheckInstrumentErrors

Exit Function

VisaComError:
MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
       Err.Source + ", " + _
       Err.Description, vbExclamation, "VISA COM Error"
End

End Function

Private Function DoQueryNumbers(query As String) As Variant()

On Error GoTo VisaComError

```

```

        Dim strErrors As String

        myScope.WriteString query
        DoQueryNumbers = myScope.ReadList
        CheckInstrumentErrors

        Exit Function

VisaComError:
        MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
            Err.Source + ", " + _
            Err.Description, vbExclamation, "VISA COM Error"
    End

End Function

Private Function DoQueryIEEEBlock_UI1(query As String) As Variant

    On Error GoTo VisaComError

    myScope.WriteString query
    DoQueryIEEEBlock_UI1 = myScope.ReadIEEEBlock(BinaryType_UI1)
    CheckInstrumentErrors

    Exit Function

VisaComError:
    MsgBox "VISA COM Error: " + vbCrLf + CStr(Err.Number) + ", " + _
        Err.Source + ", " + _
        Err.Description, vbExclamation, "VISA COM Error"
    End

End Function

Private Sub CheckInstrumentErrors()

    On Error GoTo VisaComError

    Dim strErrVal As String
    Dim strOut As String

    myScope.WriteString ":SYSTem:ERRor?"      ' Query any errors data.
    strErrVal = myScope.ReadString           ' Read: Errnum, "Error String".
    While Val(strErrVal) <> 0              ' End if find: 0, "No Error".
        strOut = strOut + "INST Error: " + strErrVal
        myScope.WriteString ":SYSTem:ERRor?"    ' Request error message.
        strErrVal = myScope.ReadString         ' Read error message.
    Wend

    If Not strOut = "" Then
        MsgBox strOut, vbExclamation, "INST Error Messages"
        myScope.FlushWrite (False)
        myScope.FlushRead
    End If

    Exit Sub

```

```

VisaComError:
    MsgBox "VISA COM Error: " + vbCrLf + Err.Description

End Sub

```

## VISA COM Example in C#

To compile and run this example in Microsoft Visual Studio 2008:

- 1** Open Visual Studio.
- 2** Create a new Visual C#, Windows, Console Application project.
- 3** Cut-and-paste the code that follows into the C# source file.
- 4** Edit the program to use the VISA address of your oscilloscope.
- 5** Add a reference to the VISA COM 5.5 Type Library:
  - a** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
  - b** Choose **Add Reference...**.
  - c** In the Add Reference dialog, select the **COM** tab.
  - d** Select **VISA COM 5.5 Type Library**; then click **OK**.
- 6** Build and run the program.

For more information, see the VISA COM Help that comes with Keysight IO Libraries Suite 16.

```

/*
 * Keysight VISA COM Example in C#
 * -----
 * This program illustrates a few commonly used programming
 * features of your Keysight oscilloscope.
 * -----
 */

using System;
using System.IO;
using System.Text;
using Ivi.Visa.Interop;
using System.Runtime.InteropServices;

namespace InfiniiVision
{
    class VisaComInstrumentApp
    {
        private static VisaComInstrument myScope;

        public static void Main(string[] args)
        {
            try
            {
                myScope = new

```

```

        VisaComInstrument ("USB0::0x0957::0x17A6::US50210029::0::INSTR"
) ;
myScope.SetTimeoutSeconds(10);

// Initialize - start from a known state.
Initialize();

// Capture data.
Capture();

// Analyze the captured waveform.
Analyze();
}

catch (System.ApplicationException err)
{
    Console.WriteLine("**** VISA COM Error : " + err.Message);
}
catch (System.SystemException err)
{
    Console.WriteLine("**** System Error Message : " + err.Message);
}
catch (System.Exception err)
{
    System.Diagnostics.Debug.Fail("Unexpected Error");
    Console.WriteLine("**** Unexpected Error : " + err.Message);
}
finally
{
    myScope.Close();
}
}

/*
 * Initialize the oscilloscope to a known state.
 * -----
 */
private static void Initialize()
{
    string strResults;

    // Get and display the device's *IDN? string.
    strResults = myScope.DoQueryString("*IDN?");
    Console.WriteLine("*IDN? result is: {0}", strResults);

    // Clear status and load the default setup.
    myScope.DoCommand("*CLS");
    myScope.DoCommand("*RST");
}

/*
 * Capture the waveform.
 * -----
 */
private static void Capture()
{
    // Use auto-scale to automatically configure oscilloscope.
    myScope.DoCommand(":AUToscale");
}

```

```

// Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
myScope.DoCommand(":TRIGger:MODE EDGE");
Console.WriteLine("Trigger mode: {0}",
    myScope.DoQueryString(":TRIGger:MODE?"));

// Set EDGE trigger parameters.
myScope.DoCommand(":TRIGger:EDGE:SOURCe CHANnel1");
Console.WriteLine("Trigger edge source: {0}",
    myScope.DoQueryString(":TRIGger:EDGE:SOURce?"));

myScope.DoCommand(":TRIGger:EDGE:LEVel 1.5");
Console.WriteLine("Trigger edge level: {0}",
    myScope.DoQueryString(":TRIGger:EDGE:LEVel?"));

myScope.DoCommand(":TRIGger:EDGE:SLOPe POSitive");
Console.WriteLine("Trigger edge slope: {0}",
    myScope.DoQueryString(":TRIGger:EDGE:SLOPe?"));

// Save oscilloscope configuration.
byte[] ResultsArray; // Results array.
int nLength; // Number of bytes returned from instrument.
string strPath;

// Query and read setup string.
ResultsArray = myScope.DoQueryIEEEBlock(":SYSTem:SETup?");
nLength = ResultsArray.Length;

// Write setup string to file.
strPath = "c:\\scope\\config\\setup.stp";
FileStream fStream = File.Open(strPath, FileMode.Create);
fStream.Write(ResultsArray, 0, nLength);
fStream.Close();
Console.WriteLine("Setup bytes saved: {0}", nLength);

// Change settings with individual commands:

// Set vertical scale and offset.
myScope.DoCommand(":CHANnel1:SCALe 0.05");
Console.WriteLine("Channel 1 vertical scale: {0}",
    myScope.DoQueryString(":CHANnel1:SCALe?"));

myScope.DoCommand(":CHANnel1:OFFSet -1.5");
Console.WriteLine("Channel 1 vertical offset: {0}",
    myScope.DoQueryString(":CHANnel1:OFFSet?"));

// Set horizontal scale and offset.
myScope.DoCommand(":TIMEbase:SCALe 0.0002");
Console.WriteLine("Timebase scale: {0}",
    myScope.DoQueryString(":TIMEbase:SCALe?"));

myScope.DoCommand(":TIMEbase:POSition 0.0");
Console.WriteLine("Timebase position: {0}",
    myScope.DoQueryString(":TIMEbase:POSition?"));

// Set the acquisition type (NORMAL, PEAK, AVERAGE, or HRESolution
).

```

```

myScope.DoCommand(":ACQuire:TYPE NORMal");
Console.WriteLine("Acquire type: {0}",
    myScope.DoQueryString(":ACQuire:TYPE?"));

// Or, configure by loading a previously saved setup.
byte[] dataArray;
int nBytesWritten;

// Read setup string from file.
strPath = "c:\\scope\\config\\setup.stp";
dataArray = File.ReadAllBytes(strPath);
nBytesWritten = dataArray.Length;

// Restore setup string.
myScope.DoCommandIEEEBlock(":SYSTem:SETup", dataArray);
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten);

// Capture an acquisition using :DIGItize.
myScope.DoCommand(":DIGItize CHANnel1");
}

/*
 * Analyze the captured waveform.
 * -----
 */
private static void Analyze()
{
    byte[] ResultsArray; // Results array.
    int nLength; // Number of bytes returned from instrument.
    string strPath;

    // Make a couple of measurements.
    // -----
    myScope.DoCommand(":MEASure:SOURce CHANnel1");
    Console.WriteLine("Measure source: {0}",
        myScope.DoQueryString(":MEASure:SOURce?"));

    double fResult;
    myScope.DoCommand(":MEASure:FREQuency");
    fResult = myScope.DoQueryNumber(":MEASure:FREQuency?");
    Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000);

    myScope.DoCommand(":MEASure:VAMplitude");
    fResult = myScope.DoQueryNumber(":MEASure:VAMplitude?");
    Console.WriteLine("Vertical amplitude: {0:F2} V", fResult);

    // Download the screen image.
    // -----
    myScope.DoCommand(":HARDcopy:INKSaver OFF");

    // Get the screen data.
    ResultsArray =
        myScope.DoQueryIEEEBlock(":DISPLAY:DATA? PNG, COLOR");
    nLength = ResultsArray.Length;

    // Store the screen data to a file.
    strPath = "c:\\scope\\data\\screen.png";
}

```

```

FileStream fStream = File.Open(strPath, FileMode.Create);
fStream.Write(ResultsArray, 0, nLength);
fStream.Close();
Console.WriteLine("Screen image ({0} bytes) written to {1}",
    nLength, strPath);

// Download waveform data.
// -----
// Set the waveform points mode.
myScope.DoCommand(":WAVEform:POINTS:MODE RAW");
Console.WriteLine("Waveform points mode: {0}",
    myScope.DoQueryString(":WAVEform:POINTS:MODE?"));

// Get the number of waveform points available.
Console.WriteLine("Waveform points available: {0}",
    myScope.DoQueryString(":WAVEform:POINTS?"));

// Set the waveform source.
myScope.DoCommand(":WAVEform:SOURce CHANnel1");
Console.WriteLine("Waveform source: {0}",
    myScope.DoQueryString(":WAVEform:SOURce?"));

// Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.DoCommand(":WAVEform:FORMAT BYTE");
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMAT?"));

// Display the waveform settings:
double[] fResultsArray;
fResultsArray = myScope.DoQueryNumbers(":WAVEform:PREamble?");

double fFormat = fResultsArray[0];
if (fFormat == 0.0)
{
    Console.WriteLine("Waveform format: BYTE");
}
else if (fFormat == 1.0)
{
    Console.WriteLine("Waveform format: WORD");
}
else if (fFormat == 2.0)
{
    Console.WriteLine("Waveform format: ASCII");
}

double fType = fResultsArray[1];
if (fType == 0.0)
{
    Console.WriteLine("Acquire type: NORMAL");
}
else if (fType == 1.0)
{
    Console.WriteLine("Acquire type: PEAK");
}
else if (fType == 2.0)
{
}

```

```

        Console.WriteLine("Acquire type: AVERage");
    }
else if (fType == 3.0)
{
    Console.WriteLine("Acquire type: HRESolution");
}

double fPoints = fResultsArray[2];
Console.WriteLine("Waveform points: {0:e}", fPoints);

double fCount = fResultsArray[3];
Console.WriteLine("Waveform average count: {0:e}", fCount);

double fXincrement = fResultsArray[4];
Console.WriteLine("Waveform X increment: {0:e}", fXincrement);

double fXorigin = fResultsArray[5];
Console.WriteLine("Waveform X origin: {0:e}", fXorigin);

double fXreference = fResultsArray[6];
Console.WriteLine("Waveform X reference: {0:e}", fXreference);

double fYincrement = fResultsArray[7];
Console.WriteLine("Waveform Y increment: {0:e}", fYincrement);

double fYorigin = fResultsArray[8];
Console.WriteLine("Waveform Y origin: {0:e}", fYorigin);

double fYreference = fResultsArray[9];
Console.WriteLine("Waveform Y reference: {0:e}", fYreference);

// Read waveform data.
ResultsArray = myScope.DoQueryIEEEBlock(":WAVEFORM:DATA?");
nLength = ResultsArray.Length;
Console.WriteLine("Number of data values: {0}", nLength);

// Set up output file:
strPath = "c:\\scope\\data\\waveform_data.csv";
if (File.Exists(strPath)) File.Delete(strPath);

// Open file for output.
StreamWriter writer = File.CreateText(strPath);

// Output waveform data in CSV format.
for (int i = 0; i < nLength - 1; i++)
    writer.WriteLine("{0:f9}, {1:f6}",
                    fXorigin + ((float)i * fXincrement),
                    (((float)ResultsArray[i] - fYreference)
                     * fYincrement) + fYorigin);

// Close output file.
writer.Close();
Console.WriteLine("Waveform format BYTE data written to {0}",
                 strPath);
}
}

```

```

class VisaComInstrument
{
    private ResourceManagerClass m_ResourceManager;
    private FormattedIO488Class m_IoObject;
    private string m_strVisaAddress;

    // Constructor.
    public VisaComInstrument(string strVisaAddress)
    {
        // Save VISA address in member variable.
        m_strVisaAddress = strVisaAddress;

        // Open the default VISA COM IO object.
        OpenIo();

        // Clear the interface.
        m_IoObject.IO.Clear();
    }

    public void DoCommand(string strCommand)
    {
        // Send the command.
        m_IoObject.WriteString(strCommand, true);

        // Check for inst errors.
        CheckInstrumentErrors(strCommand);
    }

    public void DoCommandIEEEBlock(string strCommand,
                                  byte[] DataArray)
    {
        // Send the command to the device.
        m_IoObject.WriteIEEEBlock(strCommand, DataArray, true);

        // Check for inst errors.
        CheckInstrumentErrors(strCommand);
    }

    public string DoQueryString(string strQuery)
    {
        // Send the query.
        m_IoObject.WriteString(strQuery, true);

        // Get the result string.
        string strResults;
        strResults = m_IoObject.ReadString();

        // Check for inst errors.
        CheckInstrumentErrors(strQuery);

        // Return results string.
        return strResults;
    }

    public double DoQueryNumber(string strQuery)
    {
        // Send the query.

```

```

m_IoObject.WriteString(strQuery, true);

// Get the result number.
double fResult;
fResult = (double)m_IoObject.ReadNumber(
    IEEEASCIIType.ASCIIType_R8, true);

// Check for inst errors.
CheckInstrumentErrors(strQuery);

// Return result number.
return fResult;
}

public double[] DoQueryNumbers(string strQuery)
{
    // Send the query.
    m_IoObject.WriteString(strQuery, true);

    // Get the result numbers.
    double[] fResultsArray;
    fResultsArray = (double[])m_IoObject.ReadList(
        IEEEASCIIType.ASCIIType_R8, ",;");

    // Check for inst errors.
    CheckInstrumentErrors(strQuery);

    // Return result numbers.
    return fResultsArray;
}

public byte[] DoQueryIEEEBlock(string strQuery)
{
    // Send the query.
    m_IoObject.WriteString(strQuery, true);

    // Get the results array.
    System.Threading.Thread.Sleep(2000); // Delay before reading.
    byte[] ResultsArray;
    ResultsArray = (byte[])m_IoObject.ReadIEEEBlock(
        IEEEBinaryType.BinaryType_UI1, false, true);

    // Check for inst errors.
    CheckInstrumentErrors(strQuery);

    // Return results array.
    return ResultsArray;
}

private void CheckInstrumentErrors(string strCommand)
{
    // Check for instrument errors.
    string strInstrumentError;
    bool bFirstError = true;

    do // While not "0,No error".
    {

```

```

m_IoObject.WriteString(":SYSTem:ERRor?", true);
strInstrumentError = m_IoObject.ReadString();

if (!strInstrumentError.ToString().StartsWith("+0,"))
{
    if (bFirstError)
    {
        Console.WriteLine("ERROR(s) for command '{0}': ",
                          strCommand);
        bFirstError = false;
    }
    Console.Write(strInstrumentError);
}
} while (!strInstrumentError.ToString().StartsWith("+0,"));
}

private void OpenIo()
{
    m_ResourceManager = new ResourceManagerClass();
    m_IoObject = new FormattedIO488Class();

    // Open the default VISA COM IO object.
    try
    {
        m_IoObject.IO =
            (IMessage)m_ResourceManager.Open(m_strVisaAddress,
  AccessMode.NO_LOCK, 0, "");
    }
    catch (Exception e)
    {
        Console.WriteLine("An error occurred: {0}", e.Message);
    }
}

public void SetTimeoutSeconds(int nSeconds)
{
    m_IoObject.IO.Timeout = nSeconds * 1000;
}

public void Close()
{
    try
    {
        m_IoObject.IO.Close();
    }
    catch { }

    try
    {
        Marshal.ReleaseComObject(m_IoObject);
    }
    catch { }

    try
    {
        Marshal.ReleaseComObject(m_ResourceManager);
    }
}

```

```
        catch {  
    }  
}
```

# VISA COM Example in Visual Basic .NET

To compile and run this example in Microsoft Visual Studio 2008:

- 1** Open Visual Studio.
  - 2** Create a new Visual Basic, Windows, Console Application project.
  - 3** Cut-and-paste the code that follows into the C# source file.
  - 4** Edit the program to use the VISA address of your oscilloscope.
  - 5** Add a reference to the VISA COM 5.5 Type Library:
    - a** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
    - b** Choose **Add Reference....**
    - c** In the Add Reference dialog, select the **COM** tab.
    - d** Select **VISA COM 5.5 Type Library**; then click **OK**.
    - e** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment and choose **Properties**; then, select "InfiniiVision.VisaComInstrumentApp" as the **Startup object**.
  - 6** Build and run the program.

For more information, see the VISA COM Help that comes with Keysight IO Libraries Suite 16.

```
' Keysight VISA COM Example in Visual Basic .NET
' -----
' This program illustrates a few commonly used programming
' features of your Keysight oscilloscope.
' -----
Imports System
Imports System.IO
Imports System.Text
Imports Ivi.Visa.Interop
Imports System.Runtime.InteropServices

Namespace InfiniiVision
    Class VisaComInstrumentApp
        Private Shared myScope As VisaComInstrument

        Public Shared Sub Main(ByVal args As String())
            Try
                myScope = New _
                    VisaComInstrument("USB0::0x0957::0x17A6::US50210029::0::INSTR")
            Catch ex As Exception
                Console.WriteLine(ex.Message)
            End Try
        End Sub
    End Class
End Namespace
```

```

)
myScope.SetTimeoutSeconds(10)

' Initialize - start from a known state.
Initialize()

' Capture data.
Capture()

' Analyze the captured waveform.
Analyze()

Catch err As System.ApplicationException
    Console.WriteLine("**** VISA Error Message : " + err.Message)
Catch err As System.SystemException
    Console.WriteLine("**** System Error Message : " + err.Message)
Catch err As System.Exception
    System.Diagnostics.Debug.Fail("Unexpected Error")
    Console.WriteLine("**** Unexpected Error : " + err.Message)
Finally
    myScope.Close()
End Try
End Sub

' Initialize the oscilloscope to a known state.
' -----
Private Shared Sub Initialize()
    Dim strResults As String

    ' Get and display the device's *IDN? string.
    strResults = myScope.DoQueryString("*IDN?")
    Console.WriteLine("*IDN? result is: {0}", strResults)

    ' Clear status and load the default setup.
    myScope.DoCommand("*CLS")
    myScope.DoCommand("*RST")

End Sub

' Capture the waveform.
' -----
Private Shared Sub Capture()

    ' Use auto-scale to automatically configure oscilloscope.
    myScope.DoCommand(":AUToscale")

    ' Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
    myScope.DoCommand(":TRIGger:MODE EDGE")
    Console.WriteLine("Trigger mode: {0}", _
        myScope.DoQueryString(":TRIGger:MODE?"))

    ' Set EDGE trigger parameters.
    myScope.DoCommand(":TRIGger:EDGE:SOURCe CHANnel1")
    Console.WriteLine("Trigger edge source: {0}", _
        myScope.DoQueryString(":TRIGger:EDGE:SOURce?"))

```

```

myScope.DoCommand(":TRIGger:EDGE:LEVel 1.5")
Console.WriteLine("Trigger edge level: {0}", _
    myScope.DoQueryString(":TRIGger:EDGE:LEVel?"))

myScope.DoCommand(":TRIGger:EDGE:SLOPe POSitive")
Console.WriteLine("Trigger edge slope: {0}", _
    myScope.DoQueryString(":TRIGger:EDGE:SLOPe?"))

' Save oscilloscope configuration.
Dim ResultsArray As Byte()      ' Results array.
Dim nLength As Integer         ' Number of bytes returned from inst.
Dim strPath As String
Dim fStream As FileStream

' Query and read setup string.
ResultsArray = myScope.DoQueryIEEEBlock(":SYSTem:SETup?")
nLength = ResultsArray.Length

' Write setup string to file.
strPath = "c:\scope\config\setup.stp"
fStream = File.Open(strPath, FileMode.Create)
fStream.Write(ResultsArray, 0, nLength)
fStream.Close()
Console.WriteLine("Setup bytes saved: {0}", nLength)

' Change settings with individual commands:

' Set vertical scale and offset.
myScope.DoCommand(":CHANnel1:SCALe 0.05")
Console.WriteLine("Channel 1 vertical scale: {0}", _
    myScope.DoQueryString(":CHANnel1:SCALe?"))

myScope.DoCommand(":CHANnel1:OFFSet -1.5")
Console.WriteLine("Channel 1 vertical offset: {0}", _
    myScope.DoQueryString(":CHANnel1:OFFSet?"))

' Set horizontal scale and offset.
myScope.DoCommand(":TIMEbase:SCALe 0.0002")
Console.WriteLine("Timebase scale: {0}", _
    myScope.DoQueryString(":TIMEbase:SCALe?"))

myScope.DoCommand(":TIMEbase:POSition 0.0")
Console.WriteLine("Timebase position: {0}", _
    myScope.DoQueryString(":TIMEbase:POSition?"))

' Set the acquisition type (NORMAL, PEAK, AVERAGE, or HRESolution)

myScope.DoCommand(":ACQuire:TYPE NORMAL")
Console.WriteLine("Acquire type: {0}", _
    myScope.DoQueryString(":ACQuire:TYPE?"))

' Or, configure by loading a previously saved setup.
Dim DataArray As Byte()
Dim nBytesWritten As Integer

' Read setup string from file.

```

```

strPath = "c:\scope\config\setup.stp"
DataArray = File.ReadAllBytes(strPath)
nBytesWritten = DataArray.Length

' Restore setup string.
myScope.DoCommandIEEEBlock(":SYSTem:SETUp", DataArray)
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten)

' Capture an acquisition using :DIGItize.
myScope.DoCommand(":DIGItize CHANnel1")

End Sub

' Analyze the captured waveform.
' ----

Private Shared Sub Analyze()

Dim fResult As Double
Dim ResultsArray As Byte()      ' Results array.
Dim nLength As Integer         ' Number of bytes returned from inst.
Dim strPath As String

' Make a couple of measurements.
' -----
myScope.DoCommand(":MEASure:SOURce CHANnel1")
Console.WriteLine("Measure source: {0}", _
    myScope.DoQueryString(":MEASure:SOURce?"))

myScope.DoCommand(":MEASure:FREQuency")
fResult = myScope.DoQueryNumber(":MEASure:FREQuency?")
Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000)

myScope.DoCommand(":MEASure:VAMplitude")
fResult = myScope.DoQueryNumber(":MEASure:VAMplitude?")
Console.WriteLine("Vertical amplitude: {0:F2} V", fResult)

' Download the screen image.
' -----
myScope.DoCommand(":HARDcopy:INKSaver OFF")

' Get the screen data.
ResultsArray = myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG, COLOR")
nLength = ResultsArray.Length

' Store the screen data to a file.
strPath = "c:\scope\data\screen.png"
Dim fStream As FileStream
fStream = File.Open(strPath, FileMode.Create)
fStream.Write(ResultsArray, 0, nLength)
fStream.Close()
Console.WriteLine("Screen image ({0} bytes) written to {1}", _
    nLength, strPath)

' Download waveform data.
' -----

```

```

' Set the waveform points mode.
myScope.DoCommand(":WAVEform:POINTS:MODE RAW")
Console.WriteLine("Waveform points mode: {0}", _
    myScope.DoQueryString(":WAVEform:POINTS:MODE?"))

' Get the number of waveform points available.
Console.WriteLine("Waveform points available: {0}", _
    myScope.DoQueryString(":WAVEform:POINTS?"))

' Set the waveform source.
myScope.DoCommand(":WAVEform:SOURce CHANnel1")
Console.WriteLine("Waveform source: {0}", _
    myScope.DoQueryString(":WAVEform:SOURce?"))

' Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.DoCommand(":WAVEform:FORMAT BYTE")
Console.WriteLine("Waveform format: {0}", _
    myScope.DoQueryString(":WAVEform:FORMAT?"))

' Display the waveform settings:
Dim fResultsArray As Double()
fResultsArray = myScope.DoQueryNumbers(":WAVEform:PREamble?")

Dim fFormat As Double = fResultsArray(0)
If fFormat = 0 Then
    Console.WriteLine("Waveform format: BYTE")
ElseIf fFormat = 1 Then
    Console.WriteLine("Waveform format: WORD")
ElseIf fFormat = 2 Then
    Console.WriteLine("Waveform format: ASCII")
End If

Dim fType As Double = fResultsArray(1)
If fType = 0 Then
    Console.WriteLine("Acquire type: NORMAL")
ElseIf fType = 1 Then
    Console.WriteLine("Acquire type: PEAK")
ElseIf fType = 2 Then
    Console.WriteLine("Acquire type: AVERAGE")
ElseIf fType = 3 Then
    Console.WriteLine("Acquire type: HRESolution")
End If

Dim fPoints As Double = fResultsArray(2)
Console.WriteLine("Waveform points: {0:e}", fPoints)

Dim fCount As Double = fResultsArray(3)
Console.WriteLine("Waveform average count: {0:e}", fCount)

Dim fxIncrement As Double = fResultsArray(4)
Console.WriteLine("Waveform X increment: {0:e}", fxIncrement)

Dim fxOrigin As Double = fResultsArray(5)
Console.WriteLine("Waveform X origin: {0:e}", fxOrigin)

Dim fxReference As Double = fResultsArray(6)

```

```

Console.WriteLine("Waveform X reference: {0:e}", fXreference)

Dim fYincrement As Double = fResultsArray(7)
Console.WriteLine("Waveform Y increment: {0:e}", fYincrement)

Dim fYorigin As Double = fResultsArray(8)
Console.WriteLine("Waveform Y origin: {0:e}", fYorigin)

Dim fYreference As Double = fResultsArray(9)
Console.WriteLine("Waveform Y reference: {0:e}", fYreference)

' Get the waveform data.
ResultsArray = myScope.DoQueryIEEEBlock(":WAVEFORM:DATA?")
nLength = ResultsArray.Length
Console.WriteLine("Number of data values: {0}", nLength)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"
If File.Exists(strPath) Then
    File.Delete(strPath)
End If

' Open file for output.
Dim writer As StreamWriter = File.CreateText(strPath)

' Output waveform data in CSV format.
For index As Integer = 0 To nLength - 1
    ' Write time value, voltage value.
    writer.WriteLine("{0:f9}, {1:f6}", _
                    fXorigin + (CSng(index) * fXincrement), _
                    ((CSng(ResultsArray(index)) - fYreference) * fYincrement) + fYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format BYTE data written to {0}", strPath)

End Sub

End Class

Class VisaComInstrument
    Private m_ResourceManager As ResourceManagerClass
    Private m_IoObject As FormattedIO488Class
    Private m_strVisaAddress As String

    ' Constructor.
    Public Sub New(ByVal strVisaAddress As String)

        ' Save VISA address in member variable.
        m_strVisaAddress = strVisaAddress

        ' Open the default VISA COM IO object.
        OpenIo()
    End Sub
End Class

```

```

' Clear the interface.
m_IoObject.IO.Clear()

End Sub

Public Sub DoCommand(ByVal strCommand As String)

    ' Send the command.
    m_IoObject.WriteString(strCommand, True)

    ' Check for inst errors.
    CheckInstrumentErrors(strCommand)

End Sub

Public Sub DoCommandIEEEBlock(ByVal strCommand As String, _
    ByVal DataArray As Byte())

    ' Send the command to the device.
    m_IoObject.WriteIEEEBlock(strCommand, DataArray, True)

    ' Check for inst errors.
    CheckInstrumentErrors(strCommand)

End Sub

Public Function DoQueryString(ByVal strQuery As String) As String
    ' Send the query.
    m_IoObject.WriteString(strQuery, True)

    ' Get the result string.
    Dim strResults As String
    strResults = m_IoObject.ReadString()

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return results string.
    Return strResults
End Function

Public Function DoQueryNumber(ByVal strQuery As String) As Double
    ' Send the query.
    m_IoObject.WriteString(strQuery, True)

    ' Get the result number.
    Dim fResult As Double
    fResult = _
        CDbl(m_IoObject.ReadNumber(IEEEASCIIType.ASCIIType_R8, True))

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return result number.
    Return fResult
End Function

```

```

Public Function DoQueryNumbers(ByVal strQuery As String) As _
    Double()
    ' Send the query.
    m_IoObject.WriteString(strQuery, True)

    ' Get the result numbers.
    Dim fResultsArray As Double()
    fResultsArray = _
        m_IoObject.ReadList(IEEEASCIIType.ASCIIType_R8, ", ;")

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return result numbers.
    Return fResultsArray
End Function

Public _
Function DoQueryIEEEBlock(ByVal strQuery As String) As Byte()
    ' Send the query.
    m_IoObject.WriteString(strQuery, True)

    ' Get the results array.
    System.Threading.Thread.Sleep(2000) ' Delay before reading data.
    Dim ResultsArray As Byte()
    ResultsArray = _
        m_IoObject.ReadIEEEBlock(IEEEBinaryType.BinaryType_UI1, _
        False, True)

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return results array.
    Return ResultsArray
End Function

Private Sub CheckInstrumentErrors(ByVal strCommand As String)
    ' Check for instrument errors.
    Dim strInstrumentError As String
    Dim bFirstError As Boolean = True
    Do      ' While not "0,No error".
        m_IoObject.WriteString(":SYSTem:ERRor?", True)
        strInstrumentError = m_IoObject.ReadString()

        If Not strInstrumentError.ToString().StartsWith("+0,") Then
            If bFirstError Then
                Console.WriteLine("ERROR(s) for command '{0}': {1}, _", _
                    strCommand)
                bFirstError = False
            End If
            Console.Write(strInstrumentError)
        End If
    Loop While Not strInstrumentError.ToString().StartsWith("+0,")
End Sub

Private Sub OpenIo()
    m_ResourceManager = New ResourceManagerClass()

```

```

m_IoObject = New FormattedIO488Class()

' Open the default VISA COM IO object.
Try
    m_IoObject.IO =
        DirectCast(m_ResourceManager.Open(m_strVisaAddress, _
            AccessMode.NO_LOCK, 0, ""), IMessage)
Catch e As Exception
    Console.WriteLine("An error occurred: {0}", e.Message)
End Try
End Sub

Public Sub SetTimeoutSeconds(ByVal nSeconds As Integer)
    m_IoObject.IO.Timeout = nSeconds * 1000
End Sub

Public Sub Close()
    Try
        m_IoObject.IO.Close()
    Catch
        End Try

    Try
        Marshal.ReleaseComObject(m_IoObject)
    Catch
        End Try

    Try
        Marshal.ReleaseComObject(m_ResourceManager)
    Catch
        End Try
    End Sub
End Class
End Namespace

```

## VISA COM Example in Python

You can use the Python programming language with the "comtypes" package to control Keysight oscilloscopes.

The Python language and "comtypes" package can be downloaded from the web at <http://www.python.org/> and <http://starship.python.net/crew/theller/comtypes/>, respectively.

To run this example with Python and "comtypes":

- 1** Cut-and-paste the code that follows into a file named "example.py".
- 2** Edit the program to use the VISA address of your oscilloscope.
- 3** If "python.exe" can be found via your PATH environment variable, open a Command Prompt window; then, change to the folder that contains the "example.py" file, and enter:

```
python example.py
```

```

#
# Keysight VISA COM Example in Python using "comtypes"
# ****
# This program illustrates a few commonly used programming
# features of your Keysight oscilloscope.
# ****

# Import Python modules.
# -----
import string
import time
import sys
import array

from comtypes.client import GetModule
from comtypes.client import CreateObject

# Run GetModule once to generate comtypes.gen.VisaComLib.
if not hasattr(sys, "frozen"):
    GetModule("C:\Program Files (x86)\IVI Foundation\VISA\VisaCom\
GlobMgr.dll")

import comtypes.gen.VisaComLib as VisaComLib

# Global variables (booleans: 0 = False, 1 = True).
# -----


# =====
# Initialize:
# =====
def initialize():
    # Get and display the device's *IDN? string.
    idn_string = do_query_string("*IDN?")
    print "Identification string '%s'" % idn_string

    # Clear status and load the default setup.
    do_command("*CLS")
    do_command("*RST")



# =====
# Capture:
# =====
def capture():

    # Use auto-scale to automatically set up oscilloscope.
    print "Autoscale."
    do_command(":AUToscale")

    # Set trigger mode.
    do_command(":TRIGger:MODE EDGE")
    qresult = do_query_string(":TRIGger:MODE?")
    print "Trigger mode: %s" % qresult

    # Set EDGE trigger parameters.

```

```

do_command(":TRIGger:EDGE:SOURCe CHANnel1")
qresult = do_query_string(":TRIGger:EDGE:SOURCe?")
print "Trigger edge source: %s" % qresult

do_command(":TRIGger:EDGE:LEVel 1.5")
qresult = do_query_string(":TRIGger:EDGE:LEVel?")
print "Trigger edge level: %s" % qresult

do_command(":TRIGger:EDGE:SLOPe POSitive")
qresult = do_query_string(":TRIGger:EDGE:SLOPe?")
print "Trigger edge slope: %s" % qresult

# Save oscilloscope setup.
setup_bytes = do_query_ieee_block(":SYSTem:SETUp?")
nLength = len(setup_bytes)
f = open("c:\scope\config\setup.stp", "wb")
f.write(bytarray(setup_bytes))
f.close()
print "Setup bytes saved: %d" % nLength

# Change oscilloscope settings with individual commands:

# Set vertical scale and offset.
do_command(":CHANnel1:SCALe 0.05")
qresult = do_query_number(":CHANnel1:SCALe?")
print "Channel 1 vertical scale: %f" % qresult

do_command(":CHANnel1:OFFSet -1.5")
qresult = do_query_number(":CHANnel1:OFFSet?")
print "Channel 1 offset: %f" % qresult

# Set horizontal scale and offset.
do_command(":TIMEbase:SCALe 0.0002")
qresult = do_query_string(":TIMEbase:SCALe?")
print "Timebase scale: %s" % qresult

do_command(":TIMEbase:POSItion 0.0")
qresult = do_query_string(":TIMEbase:POSItion?")
print "Timebase position: %s" % qresult

# Set the acquisition type.
do_command(":ACQuire:TYPE NORMAL")
qresult = do_query_string(":ACQuire:TYPE?")
print "Acquire type: %s" % qresult

# Or, configure by loading a previously saved setup.
f = open("c:\scope\config\setup.stp", "rb")
setup_bytes = f.read()
f.close()
do_command_ieee_block(":SYSTem:SETUp", array.array('B', setup_bytes))
print "Setup bytes restored: %d" % len(setup_bytes)

# Capture an acquisition using :DIGItize.
do_command(":DIGItize CHANnel1")

# =====

```

```

# Analyze:
# =====
def analyze():

    # Make measurements.
    # -----
    do_command(":MEASure:SOURce CHANnel1")
    qresult = do_query_string(":MEASure:SOURce?")
    print "Measure source: %s" % qresult

    do_command(":MEASure:FREQuency")
    qresult = do_query_string(":MEASure:FREQuency?")
    print "Measured frequency on channel 1: %s" % qresult

    do_command(":MEASure:VAMPplitude")
    qresult = do_query_string(":MEASure:VAMPplitude?")
    print "Measured vertical amplitude on channel 1: %s" % qresult

    # Download the screen image.
    # -----
    do_command(":HARDcopy:INKSaver OFF")

    image_bytes = do_query_ieee_block(":DISPLAY:DATA? PNG, COLOR")
    nLength = len(image_bytes)
    f = open("c:\scope\data\screen.png", "wb")
    f.write(bytarray(image_bytes))
    f.close()
    print "Screen image written to c:\scope\data\screen.png."

    # Download waveform data.
    # ----

    # Set the waveform points mode.
    do_command(":WAVeform:POINTS:MODE RAW")
    qresult = do_query_string(":WAVeform:POINTS:MODE?")
    print "Waveform points mode: %s" % qresult

    # Get the number of waveform points available.
    do_command(":WAVeform:POINTS 10240")
    qresult = do_query_string(":WAVeform:POINTS?")
    print "Waveform points available: %s" % qresult

    # Set the waveform source.
    do_command(":WAVeform:SOURce CHANnel1")
    qresult = do_query_string(":WAVeform:SOURce?")
    print "Waveform source: %s" % qresult

    # Choose the format of the data returned:
    do_command(":WAVeform:FORMAT BYTE")
    print "Waveform format: %s" % do_query_string(":WAVeform:FORMAT?")

    # Display the waveform settings from preamble:
    wav_form_dict = {
        0 : "BYTE",
        1 : "WORD",
        4 : "ASCII",
    }

```

```

acq_type_dict = {
    0 : "NORMAl",
    1 : "PEAK",
    2 : "AVERage",
    3 : "HRESolution",
}
(
    wav_form,
    acq_type,
    wfmpts,
    avgcnt,
    x_increment,
    x_origin,
    x_reference,
    y_increment,
    y_origin,
    y_reference
) = do_query_numbers(":WAVeform:PREamble?")

print "Waveform format: %s" % wav_form_dict[wav_form]
print "Acquire type: %s" % acq_type_dict[acq_type]
print "Waveform points desired: %d" % wfmpts
print "Waveform average count: %d" % avgcnt
print "Waveform X increment: %1.12f" % x_increment
print "Waveform X origin: %1.9f" % x_origin
print "Waveform X reference: %d" % x_reference # Always 0.
print "Waveform Y increment: %f" % y_increment
print "Waveform Y origin: %f" % y_origin
print "Waveform Y reference: %d" % y_reference # Always 125.

# Get numeric values for later calculations.
x_increment = do_query_number(":WAVeform:XINCrement?")
x_origin = do_query_number(":WAVeform:XORigin?")
y_increment = do_query_number(":WAVeform:YINCrement?")
y_origin = do_query_number(":WAVeform:YORigin?")
y_reference = do_query_number(":WAVeform:YREFerence?")

# Get the waveform data.
data_bytes = do_query_ieee_block(":WAVeform:DATA?")
nLength = len(data_bytes)
print "Number of data values: %d" % nLength

# Open file for output.
strPath = "c:\scope\data\waveform_data.csv"
f = open(strPath, "w")

# Output waveform data in CSV format.
for i in xrange(0, nLength - 1):
    time_val = x_origin + (i * x_increment)
    voltage = (data_bytes[i] - y_reference) * y_increment + y_origin
    f.write("%E, %f\n" % (time_val, voltage))

# Close output file.
f.close()
print "Waveform format BYTE data written to %s." % strPath

```

```

# =====
# Send a command and check for errors:
# =====
def do_command(command):
    myScope.WriteString("%s" % command, True)
    check_instrument_errors(command)

# =====
# Send a command and check for errors:
# =====
def do_command_ieee_block(command, data):
    myScope.WriteIEEEBlock(command, data, True)
    check_instrument_errors(command)

# =====
# Send a query, check for errors, return string:
# =====
def do_query_string(query):
    myScope.WriteString("%s" % query, True)
    result = myScope.ReadString()
    check_instrument_errors(query)
    return result

# =====
# Send a query, check for errors, return string:
# =====
def do_query_ieee_block(query):
    myScope.WriteString("%s" % query, True)
    result = myScope.ReadIEEEBlock(VisaComLib.BinaryType_UI1, \
        False, True)
    check_instrument_errors(query)
    return result

# =====
# Send a query, check for errors, return values:
# =====
def do_query_number(query):
    myScope.WriteString("%s" % query, True)
    result = myScope.ReadNumber(VisaComLib.ASCIIType_R8, True)
    check_instrument_errors(query)
    return result

# =====
# Send a query, check for errors, return values:
# =====
def do_query_numbers(query):
    myScope.WriteString("%s" % query, True)
    result = myScope.ReadList(VisaComLib.ASCIIType_R8, ",;")
    check_instrument_errors(query)
    return result

```

```

# =====
# Check for instrument errors:
# =====
def check_instrument_errors(command):

    while True:
        myScope.WriteString(":SYSTem:ERRor?", True)
        error_string = myScope.ReadString()
        if error_string:    # If there is an error string value.

            if error_string.find("+0,, 0, 3) == -1:    # Not "No error".
                print "ERROR: %s, command: '%s'" % (error_string, command)
                print "Exited because of error."
                sys.exit(1)

            else:    # "No error"
                break

        else:    # :SYSTem:ERRor? should always return string.
            print "ERROR: :SYSTem:ERRor? returned nothing, command: '%s' \
                  % command
            print "Exited because of error."
            sys.exit(1)

# =====
# Main program:
# =====
rm = CreateObject("VISA.GlobalRM", \
    interface=VisaComLib.IResourceManager)
myScope = CreateObject("VISA.BasicFormattedIO", \
    interface=VisaComLib.IFormattedIO488)
myScope.IO = \
    rm.Open("TCPIPO::a-mx3104a-90028.cos.keysight.com::inst0::INSTR")

# Clear the interface.
myScope.IO.Clear
print "Interface cleared."

# Set the Timeout to 15 seconds.
myScope.IO.Timeout = 15000    # 15 seconds.
print "Timeout set to 15000 milliseconds."

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

print "End of program"

```

## VISA Examples

- "VISA Example in C" on page 1377
- "VISA Example in Visual Basic" on page 1386
- "VISA Example in C#" on page 1396
- "VISA Example in Visual Basic .NET" on page 1407
- "VISA Example in Python (PyVISA 1.5 and older)" on page 1417
- "VISA Example in Python (PyVISA 1.6 and newer)" on page 1423

### VISA Example in C

To compile and run this example in Microsoft Visual Studio 2008:

- 1 Open Visual Studio.
- 2 Create a new Visual C++, Win32, Win32 Console Application project.
- 3 In the Win32 Application Wizard, click **Next >**. Then, check **Empty project**, and click **Finish**.
- 4 Cut-and-paste the code that follows into a file named "example.c" in the project directory.
- 5 In Visual Studio 2008, right-click the Source Files folder, choose **Add > Add Existing Item...**, select the example.c file, and click **Add**.
- 6 Edit the program to use the VISA address of your oscilloscope.
- 7 Choose **Project > Properties....** In the Property Pages dialog, update these project settings:
  - a Under Configuration Properties, Linker, Input, add "visa32.lib" to the Additional Dependencies field.
  - b Under Configuration Properties, C/C++, Code Generation, select Multi-threaded DLL for the Runtime Library field.
  - c Click **OK** to close the Property Pages dialog.
- 8 Add the include files and library files search paths:
  - a Choose **Tools > Options....**
  - b In the Options dialog, under Projects and Solutions, select **VC++ Directories**.
  - c Show directories for **Include files**, and add the include directory (for example, Program Files (x86)\IVI Foundation\VISA\WinNT\Include).
  - d Show directories for **Library files**, and add the library files directory (for example, Program Files (x86)\IVI Foundation\VISA\WinNT\lib\msc).
  - e Click **OK** to close the Options dialog.
- 9 Build and run the program.

```

/*
 * Keysight VISA Example in C
 * -----
 * This program illustrates a few commonly-used programming
 * features of your Keysight oscilloscope.
 */

#include <stdio.h>           /* For printf(). */
#include <string.h>          /* For strcpy(), strcat(). */
#include <time.h>            /* For clock(). */
#include <visa.h>             /* Keysight VISA routines. */

#define VISA_ADDRESS "USB0::0x0957::0x17A6::US50210029::0::INSTR"
#define IEEEBLOCK_SPACE 5000000

/* Function prototypes */
void initialize(void);           /* Initialize to known state. */
void capture(void);              /* Capture the waveform. */
void analyze(void);              /* Analyze the captured waveform. */

void do_command(char *command);   /* Send command. */
int do_command_ieeeblock(char *command); /* Command w/IEEE block. */
void do_query_string(char *query); /* Query for string. */
void do_query_number(char *query); /* Query for number. */
void do_query_numbers(char *query); /* Query for numbers. */
int do_query_ieeeblock(char *query); /* Query for IEEE block. */
void check_instrument_errors();   /* Check for inst errors. */
void error_handler();            /* VISA error handler. */

/* Global variables */
ViSession defaultRM, vi;         /* Device session ID. */
ViStatus err;                   /* VISA function return value. */
char str_result[256] = {0};       /* Result from do_query_string(). */
double num_result;              /* Result from do_query_number(). */
unsigned char ieeeblock_data[IEEEBLOCK_SPACE]; /* Result from
  do_query_ieeeblock(). */
double dbl_results[10];          /* Result from do_query_numbers(). */

/* Main Program
 * -----
void main(void)
{
    /* Open the default resource manager session. */
    err = viOpenDefaultRM(&defaultRM);
    if (err != VI_SUCCESS) error_handler();

    /* Open the session using the oscilloscope's VISA address. */
    err = viOpen(defaultRM, VISA_ADDRESS, VI_NULL, VI_NULL, &vi);
    if (err != VI_SUCCESS) error_handler();

    /* Set the I/O timeout to fifteen seconds. */
    err = viSetAttribute(vi, VI_ATTR_TMO_VALUE, 15000);
    if (err != VI_SUCCESS) error_handler();

    /* Initialize - start from a known state. */
    initialize();
}

```

```

/* Capture data. */
capture();

/* Analyze the captured waveform. */
analyze();

/* Close the vi session and the resource manager session. */
viClose(vi);
viClose(defaultRM);
}

/* Initialize the oscilloscope to a known state.
 * -----
void initialize (void)
{
    /* Clear the interface. */
    err = viClear(vi);
    if (err != VI_SUCCESS) error_handler();

    /* Get and display the device's *IDN? string. */
    do_query_string("*IDN?");
    printf("Oscilloscope *IDN? string: %s\n", str_result);

    /* Clear status and load the default setup. */
    do_command("*CLS");
    do_command("*RST");
}

/* Capture the waveform.
 * -----
void capture (void)
{
    int num_bytes;
    FILE *fp;

    /* Use auto-scale to automatically configure oscilloscope. */
    do_command(":AUToscale");

    /* Set trigger mode (EDGE, PULSe, PATTern, etc., and input source. */
    do_command(":TRIGger:MODE EDGE");
    do_query_string(":TRIGger:MODE?");
    printf("Trigger mode: %s\n", str_result);

    /* Set EDGE trigger parameters. */
    do_command(":TRIGger:EDGE:SOURce CHANnel1");
    do_query_string(":TRIGger:EDGE:SOURce?");
    printf("Trigger edge source: %s\n", str_result);

    do_command(":TRIGger:EDGE:LEVel 1.5");
    do_query_string(":TRIGger:EDGE:LEVel?");
    printf("Trigger edge level: %s\n", str_result);

    do_command(":TRIGger:EDGE:SLOPe POSitive");
    do_query_string(":TRIGger:EDGE:SLOPe?");
    printf("Trigger edge slope: %s\n", str_result);

    /* Save oscilloscope configuration. */
}

```

```

/* Read system setup. */
num_bytes = do_query_ieeeblock(":SYSTem:SEtup?");
printf("Read setup string query (%d bytes).\n", num_bytes);

/* Write setup string to file. */
fp = fopen ("c:\\scope\\config\\setup.stp", "wb");
num_bytes = fwrite(ieeeblock_data, sizeof(unsigned char), num_bytes,
    fp);
fclose (fp);
printf("Wrote setup string (%d bytes) to ", num_bytes);
printf("c:\\scope\\config\\setup.stp.\n");

/* Change settings with individual commands:

/* Set vertical scale and offset. */
do_command(":CHANnel1:SCALe 0.05");
do_query_string(":CHANnel1:SCALe?");
printf("Channel 1 vertical scale: %s\n", str_result);

do_command(":CHANnel1:OFFSet -1.5");
do_query_string(":CHANnel1:OFFSet?");
printf("Channel 1 offset: %s\n", str_result);

/* Set horizontal scale and offset. */
do_command(":TIMEbase:SCALe 0.0002");
do_query_string(":TIMEbase:SCALe?");
printf("Timebase scale: %s\n", str_result);

do_command(":TIMEbase:POSITION 0.0");
do_query_string(":TIMEbase:POSITION?");
printf("Timebase position: %s\n", str_result);

/* Set the acquisition type (NORMal, PEAK, AVERage, or HRESolution). */
do_command(":ACQuire:TYPE NORMal");
do_query_string(":ACQuire:TYPE?");
printf("Acquire type: %s\n", str_result);

/* Or, configure by loading a previously saved setup. */

/* Read setup string from file. */
fp = fopen ("c:\\scope\\config\\setup.stp", "rb");
num_bytes = fread (ieeeblock_data, sizeof(unsigned char),
    IEEEBLOCK_SPACE, fp);
fclose (fp);
printf("Read setup string (%d bytes) from file ", num_bytes);
printf("c:\\scope\\config\\setup.stp.\n");

/* Restore setup string. */
num_bytes = do_command_ieeeblock(":SYSTem:SEtup", num_bytes);
printf("Restored setup string (%d bytes).\n", num_bytes);

/* Capture an acquisition using :DIGitize. */
do_command(":DIGitize CHANnel1");
}

```

```

/* Analyze the captured waveform.
 * -----
void analyze (void)
{
    double wav_format;
    double acq_type;
    double wav_points;
    double avg_count;
    double x_increment;
    double x_origin;
    double x_reference;
    double y_increment;
    double y_origin;
    double y_reference;

    FILE *fp;
    int num_bytes; /* Number of bytes returned from instrument. */
    int i;

    /* Make a couple of measurements.
     * -----
do_command(":MEASure:SOURce CHANnel1");
do_query_string(":MEASure:SOURce?");
printf("Measure source: %s\n", str_result);

do_command(":MEASure:FREQuency");
do_query_number(":MEASure:FREQuency?");
printf("Frequency: %.4f kHz\n", num_result / 1000);

do_command(":MEASure:VAMPplitude");
do_query_number(":MEASure:VAMPplitude?");
printf("Vertical amplitude: %.2f V\n", num_result);

    /* Download the screen image.
     * -----
do_command(":HARDcopy:INKSaver OFF");

    /* Read screen image. */
num_bytes = do_query_ieeeblock(":DISPlay:DATA? PNG, COLOR");
printf("Screen image bytes: %d\n", num_bytes);

    /* Write screen image bytes to file. */
fp = fopen ("c:\\scope\\data\\screen.png", "wb");
num_bytes = fwrite(ieeeblock_data, sizeof(unsigned char), num_bytes,
    fp);
fclose (fp);
printf("Wrote screen image (%d bytes) to ", num_bytes);
printf("c:\\scope\\data\\screen.png.\n");

    /* Download waveform data.
     * -----
/* Set the waveform points mode. */
do_command(":WAVEform:POINTs:MODE RAW");
do_query_string(":WAVEform:POINTs:MODE?");
printf("Waveform points mode: %s\n", str_result);

```

```

/* Get the number of waveform points available. */
do_query_string(":WAVeform:POINTS?");
printf("Waveform points available: %s\n", str_result);

/* Set the waveform source. */
do_command(":WAVeform:SOURce CHANnel1");
do_query_string(":WAVeform:SOURce?");
printf("Waveform source: %s\n", str_result);

/* Choose the format of the data returned (WORD, BYTE, ASCII): */
do_command(":WAVeform:FORMat BYTE");
do_query_string(":WAVeform:FORMat?");
printf("Waveform format: %s\n", str_result);

/* Display the waveform settings: */
do_query_numbers(":WAVeform:PREamble?");

wav_format = dbl_results[0];
if (wav_format == 0.0)
{
    printf("Waveform format: BYTE\n");
}
else if (wav_format == 1.0)
{
    printf("Waveform format: WORD\n");
}
else if (wav_format == 2.0)
{
    printf("Waveform format: ASCii\n");
}

acq_type = dbl_results[1];
if (acq_type == 0.0)
{
    printf("Acquire type: NORMAL\n");
}
else if (acq_type == 1.0)
{
    printf("Acquire type: PEAK\n");
}
else if (acq_type == 2.0)
{
    printf("Acquire type: AVERage\n");
}
else if (acq_type == 3.0)
{
    printf("Acquire type: HRESolution\n");
}

wav_points = dbl_results[2];
printf("Waveform points: %e\n", wav_points);

avg_count = dbl_results[3];
printf("Waveform average count: %e\n", avg_count);

x_increment = dbl_results[4];
printf("Waveform X increment: %e\n", x_increment);

```

```

x_origin = dbl_results[5];
printf("Waveform X origin: %e\n", x_origin);

x_reference = dbl_results[6];
printf("Waveform X reference: %e\n", x_reference);

y_increment = dbl_results[7];
printf("Waveform Y increment: %e\n", y_increment);

y_origin = dbl_results[8];
printf("Waveform Y origin: %e\n", y_origin);

y_reference = dbl_results[9];
printf("Waveform Y reference: %e\n", y_reference);

/* Read waveform data. */
num_bytes = do_query_ieeeblock(":WAVEform:DATA?");
printf("Number of data values: %d\n", num_bytes);

/* Open file for output. */
fp = fopen("c:\\scope\\data\\waveform_data.csv", "wb");

/* Output waveform data in CSV format. */
for (i = 0; i < num_bytes - 1; i++)
{
    /* Write time value, voltage value. */
    fprintf(fp, "%9f, %6f\n",
            x_origin + ((float)i * x_increment),
            (((float)ieeeblock_data[i] - y_reference) * y_increment)
            + y_origin);
}

/* Close output file. */
fclose(fp);
printf("Waveform format BYTE data written to ");
printf("c:\\scope\\data\\waveform_data.csv.\n");
}

/* Send a command to the instrument.
 * -----
void do_command(command)
char *command;
{
    char message[80];

    strcpy(message, command);
    strcat(message, "\n");
    err = viPrintf(vi, message);
    if (err != VI_SUCCESS) error_handler();

    check_instrument_errors();
}

/* Command with IEEE definite-length block.
 * -----
int do_command_ieeeblock(command, num_bytes)

```

```

char *command;
int num_bytes;
{
    char message[80];
    int data_length;

    strcpy(message, command);
    strcat(message, "#8%08d");
    err = viPrintf(vi, message, num_bytes);
    if (err != VI_SUCCESS) error_handler();

    err = viBufWrite(vi, ieeeblock_data, num_bytes, &data_length);
    if (err != VI_SUCCESS) error_handler();

    check_instrument_errors();

    return(data_length);
}

/* Query for a string result.
 * -----
void do_query_string(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");

    err = viPrintf(vi, message);
    if (err != VI_SUCCESS) error_handler();

    err = viScanf(vi, "%t", str_result);
    if (err != VI_SUCCESS) error_handler();

    check_instrument_errors();
}

/* Query for a number result.
 * -----
void do_query_number(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");

    err = viPrintf(vi, message);
    if (err != VI_SUCCESS) error_handler();

    err = viScanf(vi, "%lf", &num_result);
    if (err != VI_SUCCESS) error_handler();

    check_instrument_errors();
}

```

```

/* Query for numbers result.
 * -----
void do_query_numbers(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");

    err = viPrintf(vi, message);
    if (err != VI_SUCCESS) error_handler();

    err = viScanf(vi, "%,10lf\n", dbl_results);
    if (err != VI_SUCCESS) error_handler();

    check_instrument_errors();
}

/* Query for an IEEE definite-length block result.
 * -----
int do_query_ieeeblock(query)
char *query;
{
    char message[80];
    int data_length;

    strcpy(message, query);
    strcat(message, "\n");
    err = viPrintf(vi, message);
    if (err != VI_SUCCESS) error_handler();

    data_length = IEEEBLOCK_SPACE;
    err = viScanf(vi, "%#b\n", &data_length, ieeeblock_data);
    if (err != VI_SUCCESS) error_handler();

    if (data_length == IEEEBLOCK_SPACE )
    {
        printf("IEEE block buffer full: ");
        printf("May not have received all data.\n");
    }

    check_instrument_errors();

    return(data_length);
}

/* Check for instrument errors.
 * -----
void check_instrument_errors()
{
    char str_err_val[256] = {0};
    char str_out[800] = "";

    err = viQueryf(vi, ":SYSTem:ERROR?\n", "%t", str_err_val);
    if (err != VI_SUCCESS) error_handler();
    while(strncmp(str_err_val, "+0,No error", 3) != 0 )

```

```

{
    strcat(str_out, ", ");
    strcat(str_out, str_err_val);
    err = viQueryf(vi, ":SYSTem:ERRor?\n", "%t", str_err_val);
    if (err != VI_SUCCESS) error_handler();
}

if (strcmp(str_out, "") != 0)
{
    printf("INST Error%s\n", str_out);
    err = viFlush(vi, VI_READ_BUF);
    if (err != VI_SUCCESS) error_handler();
    err = viFlush(vi, VI_WRITE_BUF);
    if (err != VI_SUCCESS) error_handler();
}
}

/* Handle VISA errors.
 * -----
void error_handler()
{
    char err_msg[1024] = {0};

    viStatusDesc(vi, err, err_msg);
    printf("VISA Error: %s\n", err_msg);
    if (err < VI_SUCCESS)
    {
        exit(1);
    }
}

```

## VISA Example in Visual Basic

To run this example in Visual Basic for Applications:

- 1** Start the application that provides Visual Basic for Applications (for example, Microsoft Excel).
- 2** Press ALT+F11 to launch the Visual Basic editor.
- 3** Add the visa32.bas file to your project:
  - a** Choose **File > Import File....**
  - b** Navigate to the header file, visa32.bas (installed with Keysight IO Libraries Suite and found in the Program Files (x86)\IVI Foundation\VISA\WinNT\Include), select it, and click **Open**.
- 4** Choose **Insert > Module**.
- 5** Cut-and-paste the code that follows into the editor.
- 6** Edit the program to use the VISA address of your oscilloscope, and save the changes.
- 7** Run the program.

```

'
' Keysight VISA Example in Visual Basic
' -----
' This program illustrates a few commonly-used programming
' features of your Keysight oscilloscope.
' -----
'

Option Explicit

Public err As Long      ' Error returned by VISA function calls.
Public drm As Long      ' Session to Default Resource Manager.
Public vi As Long        ' Session to instrument.

' Declare variables to hold numeric values returned by
' viVScanf/viVQueryf.
Public dblQueryResult As Double
Public Const ByteArraySize = 5000000
Public retCount As Long
Public byteArray(ByteArraySize) As Byte
Public paramsArray(2) As Long
Public Const DblArraySize = 20
Public dblArray(DblArraySize) As Double

' Declare fixed length string variable to hold string value returned
' by viVScanf/viVQueryf.
Public strQueryResult As String * 200

' For Sleep subroutine.
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'
' Main Program
' -----
'

Sub Main()

    ' Open the default resource manager session.
    err = viOpenDefaultRM(drm)
    If (err <> VI_SUCCESS) Then HandleVISAError drm

    ' Open the session using the oscilloscope's VISA address.
    err = viOpen(drm, _
                "USB0::0x0957::0x17A6::US50210029::0::INSTR", 0, 15000, vi)
    If (err <> VI_SUCCESS) Then HandleVISAError drm

    ' Set the I/O timeout to ten seconds.
    err = viSetAttribute(vi, VI_ATTR_TMO_VALUE, 10000)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    ' Initialize - start from a known state.
    Initialize

    ' Capture data.
    Capture

    ' Analyze the captured waveform.
    Analyze

```

```

' Close the vi session and the resource manager session.
err = viClose(vi)
err = viClose(drm)

End Sub

'

' Initialize the oscilloscope to a known state.
' -----
Private Sub Initialize()

    ' Clear the interface.
    err = viClear(vi)
    If Not (err = VI_SUCCESS) Then HandleVISAError vi

    ' Get and display the device's *IDN? string.
    strQueryResult = DoQueryString("*IDN?")
    MsgBox "*IDN? string: " + strQueryResult, vbOKOnly, "*IDN? Result"

    ' Clear status and load the default setup.
    DoCommand "*CLS"
    DoCommand "*RST"

End Sub

'

' Capture the waveform.
' -----
Private Sub Capture()

    ' Use auto-scale to automatically configure oscilloscope.
    ' -----
    DoCommand ":AUToscale"

    ' Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
    DoCommand ":TRIGger:MODE EDGE"
    Debug.Print "Trigger mode: " + _
        DoQueryString(":TRIGger:MODE?")

    ' Set EDGE trigger parameters.
    DoCommand ":TRIGger:EDGE:SOURCe CHANnel1"
    Debug.Print "Trigger edge source: " + _
        DoQueryString(":TRIGger:EDGE:SOURCe?")

    DoCommand ":TRIGger:EDGE:LEVel 1.5"
    Debug.Print "Trigger edge level: " + _
        DoQueryString(":TRIGger:EDGE:LEVel?")

    DoCommand ":TRIGger:EDGE:SLOPe POSitive"
    Debug.Print "Trigger edge slope: " + _
        DoQueryString(":TRIGger:EDGE:SLOPe?")

    ' Save oscilloscope configuration.
    ' -----

```

```

Dim lngSetupStringSize As Long
lngSetupStringSize = DoQueryIEEEBlock_Bytes(":SYSTem:SETup?")
Debug.Print "Setup bytes saved: " + CStr(lngSetupStringSize)

' Output setup string to a file:
Dim strPath As String
strPath = "c:\scope\config\setup.dat"
If Len(Dir(strPath)) Then
    Kill strPath      ' Remove file if it exists.
End If

' Open file for output.
Dim hFile As Long
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Dim lngI As Long
For lngI = 0 To lngSetupStringSize - 1
    Put hFile, , byteArray(lngI)      ' Write data.
Next lngI
Close hFile      ' Close file.

' Change settings with individual commands:
' -----
' Set vertical scale and offset.
DoCommand ":CHANnel1:SCALe 0.05"
Debug.Print "Channel 1 vertical scale: " + _
    DoQueryString(":CHANnel1:SCALe?")

DoCommand ":CHANnel1:OFFSet -1.5"
Debug.Print "Channel 1 vertical offset: " + _
    DoQueryString(":CHANnel1:OFFSet?")

' Set horizontal scale and position.
DoCommand ":TIMEbase:SCALe 0.0002"
Debug.Print "Timebase scale: " + _
    DoQueryString(":TIMEbase:SCALe?")

DoCommand ":TIMEbase:POSIon 0.0"
Debug.Print "Timebase position: " + _
    DoQueryString(":TIMEbase:POSIon?")

' Set the acquisition type (NORMAL, PEAK, AVERage, or HRESolution).
DoCommand ":ACQuire:TYPE NORMAL"
Debug.Print "Acquire type: " + _
    DoQueryString(":ACQuire:TYPE?")

' Or, configure by loading a previously saved setup.
' -----
strPath = "c:\scope\config\setup.dat"
Open strPath For Binary Access Read As hFile      ' Open file for input.
Dim lngSetupFileSize As Long
lngSetupFileSize = LOF(hFile)      ' Length of file.
Get hFile, , byteArray      ' Read data.
Close hFile      ' Close file.
' Write learn string back to oscilloscope using ":SYSTem:SETup"
' command:

```

```

Dim lngRestored As Long
lngRestored = DoCommandIEEEBlock(":SYSTEM:SETup", lngSetupFileSize)
Debug.Print "Setup bytes restored: " + CStr(lngRestored)

' Capture an acquisition using :DIGitize.
' -----
DoCommand ":DIGitize CHANnell"

End Sub

'

' Analyze the captured waveform.
' ----

Private Sub Analyze()

    ' Make a couple of measurements.
    ' -----
    DoCommand ":MEASure:SOURce CHANNEL1"
    Debug.Print "Measure source: " + _
        DoQueryString(":MEASure:SOURce?")

    DoCommand ":MEASure:FREQuency"
    dblQueryResult = DoQueryNumber(":MEASure:FREQuency?")
    MsgBox "Frequency:" + vbCrLf + _
        FormatNumber(dblQueryResult / 1000, 4) + " kHz"

    DoCommand ":MEASure:VAMPplitude"
    dblQueryResult = DoQueryNumber(":MEASure:VAMPplitude?")
    MsgBox "Vertical amplitude:" + vbCrLf + _
        FormatNumber(dblQueryResult, 4) + " V"

    ' Download the screen image.
    ' -----
    DoCommand ":HARDcopy:INKSaver OFF"

    ' Get screen image.
    Dim lngBlockSize As Long
    lngBlockSize = DoQueryIEEEBlock_Bytes(":DISPLAY:DATA? PNG, COLOR")
    Debug.Print "Screen image bytes: " + CStr(lngBlockSize)

    ' Save screen image to a file:
    Dim strPath As String
    strPath = "c:\scope\data\screen.png"
    If Len(Dir(strPath)) Then
        Kill strPath      ' Remove file if it exists.
    End If
    Dim hFile As Long
    hFile = FreeFile
    Open strPath For Binary Access Write Lock Write As hFile
    Dim lngI As Long
    For lngI = 0 To lngBlockSize - 1
        Put hFile, , byteArray(lngI)      ' Write data.
    Next lngI
    Close hFile      ' Close file.
    MsgBox "Screen image written to " + strPath

```

```

' Download waveform data.
'

' Set the waveform points mode.
DoCommand ":WAVeform:POINts:MODE RAW"
Debug.Print "Waveform points mode: " + _
    DoQueryString ("":WAVeform:POINts:MODE?")

' Get the number of waveform points available.
Debug.Print "Waveform points available: " + _
    DoQueryString ("":WAVeform:POINts?")

' Set the waveform source.
DoCommand ":WAVeform:SOURce CHANnel1"
Debug.Print "Waveform source: " + _
    DoQueryString ("":WAVeform:SOURce?")

' Choose the format of the data returned (WORD, BYTE, ASCII):
DoCommand ":WAVeform:FORMat BYTE"
Debug.Print "Waveform format: " + _
    DoQueryString ("":WAVeform:FORMat?")

' Display the waveform settings:
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim lngYOrigin As Long
Dim lngYReference As Long
Dim strOutput As String

Dim lngNumNumbers As Long
lngNumNumbers = DoQueryNumbers ("":WAVeform:PREamble?")

intFormat = dblArray(0)
intType = dblArray(1)
lngPoints = dblArray(2)
lngCount = dblArray(3)
dblXIncrement = dblArray(4)
dblXOrigin = dblArray(5)
lngXReference = dblArray(6)
sngYIncrement = dblArray(7)
lngYOrigin = dblArray(8)
lngYReference = dblArray(9)

If intFormat = 0 Then
    Debug.Print "Waveform format: BYTE"
ElseIf intFormat = 1 Then
    Debug.Print "Waveform format: WORD"
ElseIf intFormat = 2 Then
    Debug.Print "Waveform format: ASCII"

```

```

End If

If intType = 0 Then
    Debug.Print "Acquisition type: NORMAL"
ElseIf intType = 1 Then
    Debug.Print "Acquisition type: PEAK"
ElseIf intType = 2 Then
    Debug.Print "Acquisition type: AVERage"
ElseIf intType = 3 Then
    Debug.Print "Acquisition type: HRESolution"
End If

Debug.Print "Waveform points: " + _
FormatNumber(lngPoints, 0)

Debug.Print "Waveform average count: " + _
FormatNumber(lngCount, 0)

Debug.Print "Waveform X increment: " + _
Format(dblXIncrement, "Scientific")

Debug.Print "Waveform X origin: " + _
Format(dblXOrigin, "Scientific")

Debug.Print "Waveform X reference: " + _
FormatNumber(lngXReference, 0)

Debug.Print "Waveform Y increment: " + _
Format(sngYIncrement, "Scientific")

Debug.Print "Waveform Y origin: " + _
FormatNumber(lngYOrigin, 0)

Debug.Print "Waveform Y reference: " + _
FormatNumber(lngYReference, 0)

' Get the waveform data
Dim lngNumBytes As Long
lngNumBytes = DoQueryIEEEBlock_Bytes(":WAVEFORM:DATA?")
Debug.Print "Number of data values: " + CStr(lngNumBytes)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"

' Open file for output.
Open strPath For Output Access Write Lock Write As hFile

' Output waveform data in CSV format.
Dim lngDataValue As Long

For lngI = 0 To lngNumBytes - 1
    lngDataValue = CLng(bytArray(lngI))

    ' Write time value, voltage value.
    Print #hFile, _
        FormatNumber(dblXOrigin + (lngI * dblXIncrement), 9) + _
        ", " + _

```

```

        FormatNumber(((lngDataValue - lngYReference) _
        * sngYIncrement) + lngYOrigin)

    Next lngI

    ' Close output file.
    Close hFile      ' Close file.
    MsgBox "Waveform format BYTE data written to " + _
        "c:\scope\data\waveform_data.csv."

End Sub

Private Sub DoCommand(command As String)

    err = viVPrintf(vi, command + vbLf, 0)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    CheckInstrumentErrors

End Sub

Private Function DoCommandIEEEBlock(command As String, _
    lngBlockSize As Long)

    retCount = lngBlockSize

    Dim strCommandAndLength As String
    strCommandAndLength = command + " %#" + _
        Format(lngBlockSize) + "b"

    err = viVPrintf(vi, strCommandAndLength + vbLf, paramsArray(1))
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    DoCommandIEEEBlock = retCount

    CheckInstrumentErrors

End Function

Private Function DoQueryString(query As String) As String

    Dim strResult As String * 200

    err = viVPrintf(vi, query + vbLf, 0)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    err = viVScanf(vi, "%t", strResult)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    DoQueryString = strResult

    CheckInstrumentErrors

End Function

Private Function DoQueryNumber(query As String) As Variant

```

```

        Dim dblResult As Double

        err = viVPrintf(vi, query + vbLf, 0)
        If (err <> VI_SUCCESS) Then HandleVISAError vi

        err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblResult))
        If (err <> VI_SUCCESS) Then HandleVISAError vi

        DoQueryNumber = dblResult

        CheckInstrumentErrors

    End Function

    Private Function DoQueryNumbers(query As String) As Long

        Dim dblResult As Double

        ' Send query.
        err = viVPrintf(vi, query + vbLf, 0)
        If (err <> VI_SUCCESS) Then HandleVISAError vi

        ' Set up paramsArray for multiple parameter query returning array.
        paramsArray(0) = VarPtr(retCount)
        paramsArray(1) = VarPtr(dblArray(0))

        ' Set retCount to max number of elements array can hold.
        retCount = DblArraySize

        ' Read numbers.
        err = viVScanf(vi, "%,#lf" + vbLf, paramsArray(0))
        If (err <> VI_SUCCESS) Then HandleVISAError vi

        ' retCount is now actual number of values returned by query.
        DoQueryNumbers = retCount

        CheckInstrumentErrors

    End Function

    Private Function DoQueryIEEEBlock_Bytes(query As String) As Long

        ' Send query.
        err = viVPrintf(vi, query + vbLf, 0)
        If (err <> VI_SUCCESS) Then HandleVISAError vi

        ' Set up paramsArray for multiple parameter query returning array.
        paramsArray(0) = VarPtr(retCount)
        paramsArray(1) = VarPtr(byteArray(0))

        ' Set retCount to max number of elements array can hold.
        retCount = ByteArraySize

        ' Get unsigned integer bytes.
        err = viVScanf(vi, "%#b" + vbLf, paramsArray(0))
        If (err <> VI_SUCCESS) Then HandleVISAError vi

```

```

err = viFlush(vi, VI_READ_BUF)
If (err <> VI_SUCCESS) Then HandleVISAError vi

err = viFlush(vi, VI_WRITE_BUF)
If (err <> VI_SUCCESS) Then HandleVISAError vi

' retCount is now actual number of bytes returned by query.
DoQueryIEEEBlock_Bytes = retCount

CheckInstrumentErrors

End Function

Private Sub CheckInstrumentErrors()

On Error GoTo ErrorHandler

Dim strErrMsg As String * 200
Dim strOut As String

err = viVPrintf(vi, ":SYSTem:ERRor?" + vbLf, 0)      ' Query any errors.
If (err <> VI_SUCCESS) Then HandleVISAError vi

err = viVScanf(vi, "%t", strErrMsg)      ' Read: Errnum,"Error String".
If (err <> VI_SUCCESS) Then HandleVISAError vi

While Val(strErrMsg) <> 0                  ' End if find: 0,"No Error".
    strOut = strOut + "INST Error: " + strErrMsg

    err = viVPrintf(vi, ":SYSTem:ERRor?" + vbLf, 0)      ' Request error.
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    err = viVScanf(vi, "%t", strErrMsg)      ' Read error message.
    If (err <> VI_SUCCESS) Then HandleVISAError vi

Wend

If Not strOut = "" Then
    MsgBox strOut, vbExclamation, "INST Error Messages"

    err = viFlush(vi, VI_READ_BUF)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

    err = viFlush(vi, VI_WRITE_BUF)
    If (err <> VI_SUCCESS) Then HandleVISAError vi

End If

Exit Sub

ErrorHandler:

MsgBox "*** Error : " + Error, vbExclamation
End

End Sub

```

```

Private Sub HandleVISAError(session As Long)

    Dim strVisaErr As String * 200
    Call viStatusDesc(session, err, strVisaErr)
    MsgBox "*** VISA Error : " + strVisaErr, vbExclamation

    ' If the error is not a warning, close the session.
    If err < VI_SUCCESS Then
        If session <> 0 Then Call viClose(session)
        End
    End If

End Sub

```

## VISA Example in C#

To compile and run this example in Microsoft Visual Studio 2008:

- 1** Open Visual Studio.
- 2** Create a new Visual C#, Windows, Console Application project.
- 3** Cut-and-paste the code that follows into the C# source file.
- 4** Edit the program to use the VISA address of your oscilloscope.
- 5** Add Keysight's VISA header file to your project:
  - a** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
  - b** Click **Add** and then click **Add Existing Item...**
  - c** Navigate to the header file, visa32.cs (installed with Keysight IO Libraries Suite and found in the Program Files (x86)\IVI Foundation\VISA\WinNT\Include directory), select it, but *do not click the Open button*.
  - d** Click the down arrow to the right of the **Add** button, and choose **Add as Link**.

You should now see the file underneath your project in the Solution Explorer. It will have a little arrow icon in its lower left corner, indicating that it is a link.

- 6** Build and run the program.

For more information, see the tutorial on using VISA in Microsoft .NET in the VISA Help that comes with Keysight IO Libraries Suite 17.

```

/*
 * Keysight VISA Example in C#
 * -----
 * This program illustrates a few commonly used programming
 * features of your Keysight oscilloscope.
 * -----
 */

using System;
using System.IO;
using System.Text;

```

```

namespace InfiniiVision
{
    class VisaInstrumentApp
    {
        private static VisaInstrument myScope;

        public static void Main(string[] args)
        {
            try
            {
                myScope = new
                    VisaInstrument("USB0::0x0957::0x17A6::US50210029::0::INSTR");
                myScope.SetTimeoutSeconds(10);

                // Initialize - start from a known state.
                Initialize();

                // Capture data.
                Capture();

                // Analyze the captured waveform.
                Analyze();
            }
            catch (System.ApplicationException err)
            {
                Console.WriteLine("**** VISA Error Message : " + err.Message);
            }
            catch (System.SystemException err)
            {
                Console.WriteLine("**** System Error Message : " + err.Message);
            }
            catch (System.Exception err)
            {
                System.Diagnostics.Debug.Fail("Unexpected Error");
                Console.WriteLine("**** Unexpected Error : " + err.Message);
            }
            finally
            {
                myScope.Close();
            }
        }

        /*
         * Initialize the oscilloscope to a known state.
         * -----
         */
        private static void Initialize()
        {
            StringBuilder strResults;

            // Get and display the device's *IDN? string.
            strResults = myScope.DoQueryString("*IDN?");
            Console.WriteLine("*IDN? result is: {0}", strResults);

            // Clear status and load the default setup.
        }
    }
}

```

```

        myScope.DoCommand("*CLS");
        myScope.DoCommand("*RST");
    }

/*
 * Capture the waveform.
 * -----
 */
private static void Capture()
{
    // Use auto-scale to automatically configure oscilloscope.
    myScope.DoCommand(":AUToscale");

    // Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
    myScope.DoCommand(":TRIGger:MODE EDGE");
    Console.WriteLine("Trigger mode: {0}",
        myScope.DoQueryString(":TRIGger:MODE?"));

    // Set EDGE trigger parameters.
    myScope.DoCommand(":TRIGger:EDGE:SOURCe CHANnel1");
    Console.WriteLine("Trigger edge source: {0}",
        myScope.DoQueryString(":TRIGger:EDGE:SOURce?"));

    myScope.DoCommand(":TRIGger:EDGE:LEVel 1.5");
    Console.WriteLine("Trigger edge level: {0}",
        myScope.DoQueryString(":TRIGger:EDGE:LEVel?"));

    myScope.DoCommand(":TRIGger:EDGE:SLOPe POSitive");
    Console.WriteLine("Trigger edge slope: {0}",
        myScope.DoQueryString(":TRIGger:EDGE:SLOPe?"));

    // Save oscilloscope configuration.
    byte[] ResultsArray; // Results array.
    int nLength; // Number of bytes returned from instrument.
    string strPath;

    // Query and read setup string.
    nLength = myScope.DoQueryIEEEBlock(":SYSTem:SETUp?",
        out ResultsArray);

    // Write setup string to file.
    strPath = "c:\\scope\\config\\setup.stp";
    FileStream fStream = File.Open(strPath, FileMode.Create);
    fStream.Write(ResultsArray, 0, nLength);
    fStream.Close();
    Console.WriteLine("Setup bytes saved: {0}", nLength);

    // Change settings with individual commands:

    // Set vertical scale and offset.
    myScope.DoCommand(":CHANnel1:SCALe 0.05");
    Console.WriteLine("Channel 1 vertical scale: {0}",
        myScope.DoQueryString(":CHANnel1:SCALe?"));

    myScope.DoCommand(":CHANnel1:OFFSet -1.5");
    Console.WriteLine("Channel 1 vertical offset: {0}",
        myScope.DoQueryString(":CHANnel1:OFFSet?"));
}

```

```

// Set horizontal scale and position.
myScope.DoCommand(":TIMEbase:SCALE 0.0002");
Console.WriteLine("Timebase scale: {0}",
    myScope.DoQueryString(":TIMEbase:SCALE?"));

myScope.DoCommand(":TIMEbase:POSITION 0.0");
Console.WriteLine("Timebase position: {0}",
    myScope.DoQueryString(":TIMEbase:POSITION?"));

// Set the acquisition type (NORMAL, PEAK, AVERAge, or HRESolution
).
myScope.DoCommand(":ACQuire:TYPE NORMAL");
Console.WriteLine("Acquire type: {0}",
    myScope.DoQueryString(":ACQuire:TYPE?"));

// Or, configure by loading a previously saved setup.
byte[] dataArray;
int nBytesWritten;

// Read setup string from file.
strPath = "c:\\scope\\config\\setup.stp";
dataArray = File.ReadAllBytes(strPath);

// Restore setup string.
nBytesWritten = myScope.DoCommandIEEEBlock(":SYSTem:SETup",
    dataArray);
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten);

// Capture an acquisition using :DIGitize.
myScope.DoCommand(":DIGitize CHANnel1");
}

/*
 * Analyze the captured waveform.
 * -----
 */
private static void Analyze()
{
    byte[] ResultsArray; // Results array.
    int nLength; // Number of bytes returned from instrument.
    string strPath;

    // Make a couple of measurements.
    // -----
    myScope.DoCommand(":MEASure:SOURce CHANnel1");
    Console.WriteLine("Measure source: {0}",
        myScope.DoQueryString(":MEASure:SOURce?"));

    double fResult;
    myScope.DoCommand(":MEASure:FREQuency");
    fResult = myScope.DoQueryNumber(":MEASure:FREQuency?");
    Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000);

    myScope.DoCommand(":MEASure:VAMplitude");
    fResult = myScope.DoQueryNumber(":MEASure:VAMplitude?");
    Console.WriteLine("Vertical amplitude: {0:F2} V", fResult);
}

```

```

// Download the screen image.
// -----
myScope.DoCommand(":HARDcopy:INKSaver OFF");

// Get the screen data.
nLength = myScope.DoQueryIEEEBlock(":DISPLAY:DATA? PNG, COLOR",
    out ResultsArray);

// Store the screen data to a file.
strPath = "c:\\scope\\data\\screen.png";
FileStream fStream = File.Open(strPath, FileMode.Create);
fStream.Write(ResultsArray, 0, nLength);
fStream.Close();
Console.WriteLine("Screen image ({0} bytes) written to {1}",
    nLength, strPath);

// Download waveform data.
// -----
// Set the waveform points mode.
myScope.DoCommand(":WAVEform:POINTS:MODE RAW");
Console.WriteLine("Waveform points mode: {0}",
    myScope.DoQueryString(":WAVEform:POINTS:MODE?"));

// Get the number of waveform points available.
myScope.DoCommand(":WAVEform:POINTS 10240");
Console.WriteLine("Waveform points available: {0}",
    myScope.DoQueryString(":WAVEform:POINTS?"));

// Set the waveform source.
myScope.DoCommand(":WAVEform:SOURce CHANnel1");
Console.WriteLine("Waveform source: {0}",
    myScope.DoQueryString(":WAVEform:SOURce?"));

// Choose the format of the data returned (WORD, BYTE, ASCII).
myScope.DoCommand(":WAVEform:FORMAT BYTE");
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMAT?"));

// Display the waveform settings:
double[] fResultsArray;
fResultsArray = myScope.DoQueryNumbers(":WAVEform:PREamble?");

double fFormat = fResultsArray[0];
if (fFormat == 0.0)
{
    Console.WriteLine("Waveform format: BYTE");
}
else if (fFormat == 1.0)
{
    Console.WriteLine("Waveform format: WORD");
}
else if (fFormat == 2.0)
{
    Console.WriteLine("Waveform format: ASCII");
}

```

```

double fType = fResultsArray[1];
if (fType == 0.0)
{
    Console.WriteLine("Acquire type: NORMAl");
}
else if (fType == 1.0)
{
    Console.WriteLine("Acquire type: PEAK");
}
else if (fType == 2.0)
{
    Console.WriteLine("Acquire type: AVERage");
}
else if (fType == 3.0)
{
    Console.WriteLine("Acquire type: HRESolution");
}

double fPoints = fResultsArray[2];
Console.WriteLine("Waveform points: {0:e}", fPoints);

double fCount = fResultsArray[3];
Console.WriteLine("Waveform average count: {0:e}", fCount);

double fXincrement = fResultsArray[4];
Console.WriteLine("Waveform X increment: {0:e}", fXincrement);

double fXorigin = fResultsArray[5];
Console.WriteLine("Waveform X origin: {0:e}", fXorigin);

double fXreference = fResultsArray[6];
Console.WriteLine("Waveform X reference: {0:e}", fXreference);

double fYincrement = fResultsArray[7];
Console.WriteLine("Waveform Y increment: {0:e}", fYincrement);

double fYorigin = fResultsArray[8];
Console.WriteLine("Waveform Y origin: {0:e}", fYorigin);

double fYreference = fResultsArray[9];
Console.WriteLine("Waveform Y reference: {0:e}", fYreference);

// Read waveform data.
nLength = myScope.DoQueryIEEEBlock(":WAVEform:DATA?",
    out ResultsArray);
Console.WriteLine("Number of data values: {0}", nLength);

// Set up output file:
strPath = "c:\\scope\\data\\waveform_data.csv";
if (File.Exists(strPath)) File.Delete(strPath);

// Open file for output.
StreamWriter writer = File.CreateText(strPath);

// Output waveform data in CSV format.
for (int i = 0; i < nLength - 1; i++)

```

```

        writer.WriteLine("{0:f9}, {1:f6}",
            fXorigin + ((float)i * fXincrement),
            (((float)ResultsArray[i] - fYreference) *
            fYincrement) + fYorigin);

        // Close output file.
        writer.Close();
        Console.WriteLine("Waveform format BYTE data written to {0}",
            strPath);
    }
}

class VisaInstrument
{
    private int m_nResourceManager;
    private int m_nSession;
    private string m_strVisaAddress;

    // Constructor.
    public VisaInstrument(string strVisaAddress)
    {
        // Save VISA address in member variable.
        m_strVisaAddress = strVisaAddress;

        // Open the default VISA resource manager.
        OpenResourceManager();

        // Open a VISA resource session.
        OpenSession();

        // Clear the interface.
        int nViStatus;
        nViStatus = visa32.viClear(m_nSession);
    }

    public void DoCommand(string strCommand)
    {
        // Send the command.
        VisaSendCommandOrQuery(strCommand);

        // Check for inst errors.
        CheckInstrumentErrors(strCommand);
    }

    public int DoCommandIEEEBlock(string strCommand,
        byte[] dataArray)
    {
        // Send the command to the device.
        string strCommandAndLength;
        int nViStatus, nLength, nBytesWritten;

        nLength = dataArray.Length;
        strCommandAndLength = String.Format("{0} #8%08d",
            strCommand);

        // Write first part of command to formatted I/O write buffer.
        nViStatus = visa32.viPrintf(m_nSession, strCommandAndLength,

```

```

        nLength);
CheckVisaStatus(nViStatus);

// Write the data to the formatted I/O write buffer.
nViStatus = visa32.viBufWrite(m_nSession, DataArray, nLength,
    out nBytesWritten);
CheckVisaStatus(nViStatus);

// Check for inst errors.
CheckInstrumentErrors(strCommand);

return nBytesWritten;
}

public StringBuilder DoQueryString(string strQuery)
{
    // Send the query.
    VisaSendCommandOrQuery(strQuery);

    // Get the result string.
    StringBuilder strResults = new StringBuilder(1000);
    strResults = VisaGetResultString();

    // Check for inst errors.
    CheckInstrumentErrors(strQuery);

    // Return string results.
    return strResults;
}

public double DoQueryNumber(string strQuery)
{
    // Send the query.
    VisaSendCommandOrQuery(strQuery);

    // Get the result string.
    double fResults;
    fResults = VisaGetResultNumber();

    // Check for inst errors.
    CheckInstrumentErrors(strQuery);

    // Return string results.
    return fResults;
}

public double[] DoQueryNumbers(string strQuery)
{
    // Send the query.
    VisaSendCommandOrQuery(strQuery);

    // Get the result string.
    double[] fResultsArray;
    fResultsArray = VisaGetResultNumbers();

    // Check for inst errors.
    CheckInstrumentErrors(strQuery);
}

```

```

        // Return string results.
        return fResultsArray;
    }

    public int DoQueryIEEEBlock(string strQuery,
        out byte[] ResultsArray)
    {
        // Send the query.
        VisaSendCommandOrQuery(strQuery);

        // Get the result string.
        int length;    // Number of bytes returned from instrument.
        length = VisaGetResultIEEEBlock(out ResultsArray);

        // Check for inst errors.
        CheckInstrumentErrors(strQuery);

        // Return string results.
        return length;
    }

    private void VisaSendCommandOrQuery(string strCommandOrQuery)
    {
        // Send command or query to the device.
        string strWithNewline;
        strWithNewline = String.Format("{0}\n", strCommandOrQuery);
        int nViStatus;
        nViStatus = visa32.viPrintf(m_nSession, strWithNewline);
        CheckVisaStatus(nViStatus);
    }

    private StringBuilder VisaGetString()
    {
        StringBuilder strResults = new StringBuilder(1000);

        // Read return value string from the device.
        int nViStatus;
        nViStatus = visa32.viScanf(m_nSession, "%1000t", strResults);
        CheckVisaStatus(nViStatus);

        return strResults;
    }

    private double VisaGetResultNumber()
    {
        double fResults = 0;

        // Read return value string from the device.
        int nViStatus;
        nViStatus = visa32.viScanf(m_nSession, "%lf", out fResults);
        CheckVisaStatus(nViStatus);

        return fResults;
    }

    private double[] VisaGetResultNumbers()

```

```

{
    double[] fResultsArray;
    fResultsArray = new double[10];

    // Read return value string from the device.
    int nViStatus;
    nViStatus = visa32.viScanf(m_nSession, "%,10lf\n",
        fResultsArray);
    CheckVisaStatus(nViStatus);

    return fResultsArray;
}

private int VisaGetResultIEEEBlock(out byte[] ResultsArray)
{
    // Results array, big enough to hold a PNG.
    ResultsArray = new byte[300000];
    int length; // Number of bytes returned from instrument.

    // Set the default number of bytes that will be contained in
    // the ResultsArray to 300,000 (300kB).
    length = 300000;

    // Read return value string from the device.
    int nViStatus;
    nViStatus = visa32.viScanf(m_nSession, "%#b", ref length,
        ResultsArray);
    CheckVisaStatus(nViStatus);

    // Write and read buffers need to be flushed after IEEE block?
    nViStatus = visa32.viFlush(m_nSession, visa32.VI_WRITE_BUF);
    CheckVisaStatus(nViStatus);

    nViStatus = visa32.viFlush(m_nSession, visa32.VI_READ_BUF);
    CheckVisaStatus(nViStatus);

    return length;
}

private void CheckInstrumentErrors(string strCommand)
{
    // Check for instrument errors.
    StringBuilder strInstrumentError = new StringBuilder(1000);
    bool bFirstError = true;

    do // While not "0,No error"
    {
        VisaSendCommandOrQuery(":SYSTem:ERRor?");
        strInstrumentError = VisaGetString();
        if (!strInstrumentError.ToString().StartsWith("+0,"))
        {
            if (bFirstError)
            {
                Console.WriteLine("ERROR(s) for command '{0}': {1}",
                    strCommand);
                bFirstError = false;
            }
        }
    }
}

```

```

        }
        Console.WriteLine(strInstrumentError);
    }

} while (!strInstrumentError.ToString().StartsWith("+0,"));
}

private void OpenResourceManager()
{
    int nViStatus;
    nViStatus =
        visa32.viOpenDefaultRM(out this.m_nResourceManager);
    if (nViStatus < visa32.VI_SUCCESS)
        throw new
            ApplicationException("Failed to open Resource Manager");
}

private void OpenSession()
{
    int nViStatus;
    nViStatus = visa32.viOpen(this.m_nResourceManager,
        this.m_strVisaAddress, visa32.VI_NO_LOCK,
        visa32.VI_TMO_IMMEDIATE, out this.m_nSession);
    CheckVisaStatus(nViStatus);
}

public void SetTimeoutSeconds(int nSeconds)
{
    int nViStatus;
    nViStatus = visa32.viSetAttribute(this.m_nSession,
        visa32.VI_ATTR_TMO_VALUE, nSeconds * 1000);
    CheckVisaStatus(nViStatus);
}

public void CheckVisaStatus(int nViStatus)
{
    // If VISA error, throw exception.
    if (nViStatus < visa32.VI_SUCCESS)
    {
        StringBuilder strError = new StringBuilder(256);
        visa32.viStatusDesc(this.m_nResourceManager, nViStatus,
            strError);
        throw new ApplicationException(strError.ToString());
    }
}

public void Close()
{
    if (m_nSession != 0)
        visa32.viClose(m_nSession);
    if (m_nResourceManager != 0)
        visa32.viClose(m_nResourceManager);
}
}

```

## VISA Example in Visual Basic .NET

To compile and run this example in Microsoft Visual Studio 2008:

- 1** Open Visual Studio.
- 2** Create a new Visual Basic, Windows, Console Application project.
- 3** Cut-and-paste the code that follows into the Visual Basic .NET source file.
- 4** Edit the program to use the VISA address of your oscilloscope.
- 5** Add Keysight's VISA header file to your project:
  - a** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
  - b** Choose **Add** and then choose **Add Existing Item...**
  - c** Navigate to the header file, visa32.vb (installed with Keysight IO Libraries Suite and found in the Program Files (x86)\IVI Foundation\VISA\WinNT\Include directory), select it, but *do not click the Open button*.
  - d** Click the down arrow to the right of the **Add** button, and choose **Add as Link**.

You should now see the file underneath your project in the Solution Explorer. It will have a little arrow icon in its lower left corner, indicating that it is a link.

- e** Right-click the project again and choose **Properties**; then, select "InfiniiVision.VisalnstrumentApp" as the **Startup object**.

- 6** Build and run the program.

For more information, see the tutorial on using VISA in Microsoft .NET in the VISA Help that comes with Keysight IO Libraries Suite 17.

```
'-----'
' Keysight VISA Example in Visual Basic .NET
' -----
' This program illustrates a few commonly-used programming
' features of your Keysight oscilloscope.
' -----'

Imports System
Imports System.IO
Imports System.Text

Namespace InfiniiVision
    Class VisaInstrumentApp
        Private Shared myScope As VisaInstrument

        Public Shared Sub Main(ByVal args As String())
            Try
                myScope =
                    New VisaInstrument("USB0::0x0957::0x17A6::US50210029::0::INSTR")
            myScope.SetTimeoutSeconds(10)
        End Sub
    End Class
End Namespace
```

```

' Initialize - start from a known state.
Initialize()

' Capture data.
Capture()

' Analyze the captured waveform.
Analyze()

Catch err As System.ApplicationException
    Console.WriteLine("**** VISA Error Message : " + err.Message)
Catch err As System.SystemException
    Console.WriteLine("**** System Error Message : " + err.Message)
Catch err As System.Exception
    Debug.Fail("Unexpected Error")
    Console.WriteLine("**** Unexpected Error : " + err.Message)
End Try
End Sub

'
' Initialize the oscilloscope to a known state.
' -----
Private Shared Sub Initialize()
    Dim strResults As StringBuilder

    ' Get and display the device's *IDN? string.
    strResults = myScope.DoQueryString("*IDN?")
    Console.WriteLine("*IDN? result is: {0}", strResults)

    ' Clear status and load the default setup.
    myScope.DoCommand("*CLS")
    myScope.DoCommand("*RST")

End Sub

'
' Capture the waveform.
' -----
Private Shared Sub Capture()

    ' Use auto-scale to automatically configure oscilloscope.
    myScope.DoCommand(":AUToscale")

    ' Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
    myScope.DoCommand(":TRIGger:MODE EDGE")
    Console.WriteLine("Trigger mode: {0}", _
        myScope.DoQueryString(":TRIGger:MODE?"))

    ' Set EDGE trigger parameters.
    myScope.DoCommand(":TRIGger:EDGE:SOURCe CHANnel1")
    Console.WriteLine("Trigger edge source: {0}", _
        myScope.DoQueryString(":TRIGger:EDGE:SOURce?"))

    myScope.DoCommand(":TRIGger:EDGE:LEVel 1.5")
    Console.WriteLine("Trigger edge level: {0}", _

```

```

myScope.DoQueryString(":TRIGger:EDGE:LEVel?"))

myScope.DoCommand(":TRIGger:EDGE:SLOPe POSitive")
Console.WriteLine("Trigger edge slope: {0}", _
    myScope.DoQueryString(":TRIGger:EDGE:SLOPe?"))

' Save oscilloscope configuration.
Dim ResultsArray As Byte()      ' Results array.
Dim nLength As Integer         ' Number of bytes returned from inst.
Dim strPath As String
Dim fStream As FileStream

' Query and read setup string.
nLength = myScope.DoQueryIEEEBlock(":SYSTem:SETUp?", _
    ResultsArray)

' Write setup string to file.
strPath = "c:\scope\config\setup.stp"
fStream = File.Open(strPath, FileMode.Create)
fStream.Write(ResultsArray, 0, nLength)
fStream.Close()
Console.WriteLine("Setup bytes saved: {0}", nLength)

' Change settings with individual commands:

' Set vertical scale and offset.
myScope.DoCommand(":CHANnel1:SCALe 0.05")
Console.WriteLine("Channel 1 vertical scale: {0}", _
    myScope.DoQueryString(":CHANnel1:SCALe?"))

myScope.DoCommand(":CHANnel1:OFFSet -1.5")
Console.WriteLine("Channel 1 vertical offset: {0}", _
    myScope.DoQueryString(":CHANnel1:OFFSet?"))

' Set horizontal scale and position.
myScope.DoCommand(":TIMEbase:SCALe 0.0002")
Console.WriteLine("Timebase scale: {0}", _
    myScope.DoQueryString(":TIMEbase:SCALe?"))

myScope.DoCommand(":TIMEbase:POSItion 0.0")
Console.WriteLine("Timebase position: {0}", _
    myScope.DoQueryString(":TIMEbase:POSItion?"))

' Set the acquisition type (NORMAL, PEAK, AVERAGE, or HRESolution)

myScope.DoCommand(":ACQuire:TYPE NORMAL")
Console.WriteLine("Acquire type: {0}", _
    myScope.DoQueryString(":ACQuire:TYPE?"))

' Or, configure by loading a previously saved setup.
Dim DataArray As Byte()
Dim nBytesWritten As Integer

' Read setup string from file.
strPath = "c:\scope\config\setup.stp"
DataArray = File.ReadAllBytes(strPath)

```

```

' Restore setup string.
nBytesWritten = myScope.DoCommandIEEEBlock(":SYSTem:SETup", _
    DataArray)
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten)

' Capture an acquisition using :DIGitize.
myScope.DoCommand(":DIGitize CHANnel1")

End Sub

'

' Analyze the captured waveform.
' -----
Private Shared Sub Analyze()

    Dim fResult As Double
    Dim ResultsArray As Byte()      ' Results array.
    Dim nLength As Integer        ' Number of bytes returned from inst.
    Dim strPath As String

    ' Make a couple of measurements.
    ' -----
    myScope.DoCommand(":MEASure:SOURce CHANnel1")
    Console.WriteLine("Measure source: {0}", _
        myScope.DoQueryString(":MEASure:SOURce?"))

    myScope.DoCommand(":MEASure:FREQuency")
    fResult = myScope.DoQueryNumber(":MEASure:FREQuency?")
    Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000)

    myScope.DoCommand(":MEASure:VAMplitude")
    fResult = myScope.DoQueryNumber(":MEASure:VAMplitude?")
    Console.WriteLine("Vertical amplitude: {0:F2} V", fResult)

    ' Download the screen image.
    ' -----
    myScope.DoCommand(":HARDcopy:INKSaver OFF")

    ' Get the screen data.
    nLength = myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG, COLOR", _
        ResultsArray)

    ' Store the screen data to a file.
    strPath = "c:\scope\data\screen.png"
    Dim fStream As FileStream
    fStream = File.Open(strPath, FileMode.Create)
    fStream.Write(ResultsArray, 0, nLength)
    fStream.Close()
    Console.WriteLine("Screen image ({0} bytes) written to {1}", _
        nLength, strPath)

    ' Download waveform data.
    ' -----
    ' Set the waveform points mode.
    myScope.DoCommand(":WAVeform:POINTS:MODE RAW")

```

```

Console.WriteLine("Waveform points mode: {0}", _
    myScope.DoQueryString(":WAVEform:POINts:MODE?"))

' Get the number of waveform points available.
myScope.DoCommand(":WAVEform:POINts 10240")
Console.WriteLine("Waveform points available: {0}", _
    myScope.DoQueryString(":WAVEform:POINts?"))

' Set the waveform source.
myScope.DoCommand(":WAVEform:SOURce CHANnel1")
Console.WriteLine("Waveform source: {0}", _
    myScope.DoQueryString(":WAVEform:SOURce?"))

' Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.DoCommand(":WAVEform:FORMAT BYTE")
Console.WriteLine("Waveform format: {0}", _
    myScope.DoQueryString(":WAVEform:FORMAT?"))

' Display the waveform settings:
Dim fResultsArray As Double()
fResultsArray = myScope.DoQueryNumbers(":WAVEform:PREamble?")

Dim fFormat As Double = fResultsArray(0)
If fFormat = 0 Then
    Console.WriteLine("Waveform format: BYTE")
ElseIf fFormat = 1 Then
    Console.WriteLine("Waveform format: WORD")
ElseIf fFormat = 2 Then
    Console.WriteLine("Waveform format: ASCII")
End If

Dim fType As Double = fResultsArray(1)
If fType = 0 Then
    Console.WriteLine("Acquire type: NORMAL")
ElseIf fType = 1 Then
    Console.WriteLine("Acquire type: PEAK")
ElseIf fType = 2 Then
    Console.WriteLine("Acquire type: AVERAGE")
ElseIf fType = 3 Then
    Console.WriteLine("Acquire type: HRESolution")
End If

Dim fPoints As Double = fResultsArray(2)
Console.WriteLine("Waveform points: {0:e}", fPoints)

Dim fCount As Double = fResultsArray(3)
Console.WriteLine("Waveform average count: {0:e}", fCount)

Dim fXincrement As Double = fResultsArray(4)
Console.WriteLine("Waveform X increment: {0:e}", fXincrement)

Dim fXorigin As Double = fResultsArray(5)
Console.WriteLine("Waveform X origin: {0:e}", fXorigin)

Dim fXreference As Double = fResultsArray(6)
Console.WriteLine("Waveform X reference: {0:e}", fXreference)

```

```

Dim fYincrement As Double = fResultsArray(7)
Console.WriteLine("Waveform Y increment: {0:e}", fYincrement)

Dim fYorigin As Double = fResultsArray(8)
Console.WriteLine("Waveform Y origin: {0:e}", fYorigin)

Dim fYreference As Double = fResultsArray(9)
Console.WriteLine("Waveform Y reference: {0:e}", fYreference)

' Get the waveform data.
nLength = myScope.DoQueryIEEEBlock(":WAVEform:DATA?", _
    ResultsArray)
Console.WriteLine("Number of data values: {0}", nLength)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"
If File.Exists(strPath) Then
    File.Delete(strPath)
End If

' Open file for output.
Dim writer As StreamWriter = File.CreateText(strPath)

' Output waveform data in CSV format.
For index As Integer = 0 To nLength - 1
    ' Write time value, voltage value.
    writer.WriteLine("{0:f9}, {1:f6}", _
        fXorigin + (CSng(index) * fxincrement), _
        ((CSng(ResultsArray(index)) - fYreference) _ 
         * fYincrement) + fYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format BYTE data written to {0}", _
    strPath)

End Sub

End Class

Class VisaInstrument
    Private m_nResourceManager As Integer
    Private m_nSession As Integer
    Private m_strVisaAddress As String

    ' Constructor.
    Public Sub New(ByVal strVisaAddress As String)
        ' Save VISA address in member variable.
        m_strVisaAddress = strVisaAddress

        ' Open the default VISA resource manager.
        OpenResourceManager()

        ' Open a VISA resource session.
        OpenSession()
    End Sub
End Class

```

```

' Clear the interface.
Dim nViStatus As Integer
nViStatus = visa32.viClear(m_nSession)
End Sub

Public Sub DoCommand(ByVal strCommand As String)
    ' Send the command.
    VisaSendCommandOrQuery(strCommand)

    ' Check for inst errors.
    CheckInstrumentErrors(strCommand)

End Sub

Public Function DoCommandIEEEBlock(ByVal strCommand As String, _
    ByVal DataArray As Byte()) As Integer

    ' Send the command to the device.
    Dim strCommandAndLength As String
    Dim nViStatus As Integer
    Dim nLength As Integer
    Dim nBytesWritten As Integer

    nLength = DataArray.Length
    strCommandAndLength = [String].Format("{0} #8{1:D8}", _
        strCommand, nLength)

    ' Write first part of command to formatted I/O write buffer.
    nViStatus = visa32.viPrintf(m_nSession, strCommandAndLength)
    CheckVisaStatus(nViStatus)

    ' Write the data to the formatted I/O write buffer.
    nViStatus = visa32.viBufWrite(m_nSession, DataArray, nLength, _
        nBytesWritten)
    CheckVisaStatus(nViStatus)

    ' Check for inst errors.
    CheckInstrumentErrors(strCommand)

    Return nBytesWritten
End Function

Public Function DoQueryString(ByVal strQuery As String) _
    As StringBuilder
    ' Send the query.
    VisaSendCommandOrQuery(strQuery)

    ' Get the result string.
    Dim strResults As New StringBuilder(1000)
    strResults = VisaGetResultString()

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return string results.
    Return strResults
End Function

```

```

Public Function DoQueryNumber(ByVal strQuery As String) As Double
    ' Send the query.
    VisaSendCommandOrQuery(strQuery)

    ' Get the result string.
    Dim fResults As Double
    fResults = VisaGetResultNumber()

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return string results.
    Return fResults
End Function

Public Function DoQueryNumbers(ByVal strQuery As String) _
    As Double()
    ' Send the query.
    VisaSendCommandOrQuery(strQuery)

    ' Get the result string.
    Dim fResultsArray As Double()
    fResultsArray = VisaGetResultNumbers()

    ' Check for instrument errors (another command and result).
    CheckInstrumentErrors(strQuery)

    ' Return string results.
    Return fResultsArray
End Function

Public Function DoQueryIEEEBlock(ByVal strQuery As String, _
    ByRef ResultsArray As Byte()) As Integer
    ' Send the query.
    VisaSendCommandOrQuery(strQuery)

    ' Get the result string.
    System.Threading.Thread.Sleep(2000) ' Delay before reading data.
    Dim length As Integer
    ' Number of bytes returned from instrument.
    length = VisaGetResultIEEEBlock(ResultsArray)

    ' Check for inst errors.
    CheckInstrumentErrors(strQuery)

    ' Return string results.
    Return length
End Function

Private Sub VisaSendCommandOrQuery(ByVal strCommandOrQuery _
    As String)
    ' Send command or query to the device.
    Dim strWithNewline As String
    strWithNewline = [String].Format("{0}" & Chr(10) & "", _
        strCommandOrQuery)
    Dim nViStatus As Integer

```

```

nViStatus = visa32.viPrintf(m_nSession, strWithNewline)
CheckVisaStatus(nViStatus)
End Sub

Private Function VisaGetString() As StringBuilder
    Dim strResults As New StringBuilder(1000)

    ' Read return value string from the device.
    Dim nViStatus As Integer
    nViStatus = visa32.viScanf(m_nSession, "%1000t", strResults)
    CheckVisaStatus(nViStatus)

    Return strResults
End Function

Private Function VisaGetResultNumber() As Double
    Dim fResults As Double = 0

    ' Read return value string from the device.
    Dim nViStatus As Integer
    nViStatus = visa32.viScanf(m_nSession, "%lf", fResults)
    CheckVisaStatus(nViStatus)

    Return fResults
End Function

Private Function VisaGetResultNumbers() As Double()
    Dim fResultsArray As Double()
    fResultsArray = New Double(9) {}

    ' Read return value string from the device.
    Dim nViStatus As Integer
    nViStatus = visa32.viScanf(m_nSession,
        "%,10lf" & Chr(10) & "", fResultsArray)
    CheckVisaStatus(nViStatus)

    Return fResultsArray
End Function

Private Function VisaGetResultIEEEBlock(ByRef ResultsArray _
    As Byte()) As Integer
    ' Results array, big enough to hold a PNG.
    ResultsArray = New Byte(299999) {}
    Dim length As Integer
    ' Number of bytes returned from instrument.
    ' Set the default number of bytes that will be contained in
    ' the ResultsArray to 300,000 (300kB).
    length = 300000

    ' Read return value string from the device.
    Dim nViStatus As Integer
    nViStatus = visa32.viScanf(m_nSession, "%#b", length, _
        ResultsArray)
    CheckVisaStatus(nViStatus)

    ' Write and read buffers need to be flushed after IEEE block?
    nViStatus = visa32.viFlush(m_nSession, visa32.VI_WRITE_BUF)

```

```

CheckVisaStatus(nViStatus)

nViStatus = visa32.viFlush(m_nSession, visa32.VI_READ_BUF)
CheckVisaStatus(nViStatus)

    Return length
End Function

Private Sub CheckInstrumentErrors(ByVal strCommand As String)
    ' Check for instrument errors.
    Dim strInstrumentError As New StringBuilder(1000)
    Dim bFirstError As Boolean = True
    Do    ' While not "0,No error"
        VisaSendCommandOrQuery(":SYSTem:ERRor?")
        strInstrumentError = VisaGetResultString()

        If Not strInstrumentError.ToString().StartsWith("+0,") Then
            If bFirstError Then
                Console.WriteLine("ERROR(s) for command '{0}': ", _
                    strCommand)
                bFirstError = False
            End If
            Console.Write(strInstrumentError)
        End If
    Loop While Not strInstrumentError.ToString().StartsWith("+0,")
End Sub

Private Sub OpenResourceManager()
    Dim nViStatus As Integer
    nViStatus = visa32.viOpenDefaultRM(Me.m_nResourceManager)
    If nViStatus < visa32.VI_SUCCESS Then
        Throw New _
            ApplicationException("Failed to open Resource Manager")
    End If
End Sub

Private Sub OpenSession()
    Dim nViStatus As Integer
    nViStatus = visa32.viOpen(Me.m_nResourceManager, _
        Me.m_strVisaAddress, visa32.VI_NO_LOCK, _
        visa32.VI_TMO_IMMEDIATE, Me.m_nSession)
    CheckVisaStatus(nViStatus)
End Sub

Public Sub SetTimeoutSeconds(ByVal nSeconds As Integer)
    Dim nViStatus As Integer
    nViStatus = visa32.viSetAttribute(Me.m_nSession, _
        visa32.VI_ATTR_TMO_VALUE, nSeconds * 1000)
    CheckVisaStatus(nViStatus)
End Sub

Public Sub CheckVisaStatus(ByVal nViStatus As Integer)
    ' If VISA error, throw exception.
    If nViStatus < visa32.VI_SUCCESS Then
        Dim strError As New StringBuilder(256)
        visa32.viStatusDesc(Me.m_nResourceManager, nViStatus, strError)
        Throw New ApplicationException(strError.ToString())
    End If
End Sub

```

```

        End If
    End Sub

    Public Sub Close()
        If m_nSession <> 0 Then
            visa32.viClose(m_nSession)
        End If
        If m_nResourceManager <> 0 Then
            visa32.viClose(m_nResourceManager)
        End If
    End Sub
End Class
End Namespace

```

## VISA Example in Python (PyVISA 1.5 and older)

You can use the Python programming language with the PyVISA package to control Keysight Infinium Series oscilloscopes.

The Python language and PyVISA package can be downloaded from the web at <http://www.python.org/> and <http://pyvisa.sourceforge.net/>, respectively.

To run this example with Python and PyVISA:

- 1 Cut-and-paste the code that follows into a file named "example.py".
- 2 Edit the program to use the VISA address of your oscilloscope.
- 3 If "python.exe" can be found via your PATH environment variable, open a Command Prompt window; then, change to the folder that contains the "example.py" file, and enter:

```

python example.py

# ****
# This program illustrates a few commonly-used programming
# features of your Keysight oscilloscope.
# ****

# Import modules.
# -----
import visa
import string
import struct
import sys

# Global variables (booleans: 0 = False, 1 = True).
# -----
debug = 0

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.

```

```

idn_string = do_query_string("*IDN?")
print "Identification string: '%s'" % idn_string

# Clear status and load the default setup.
do_command("*CLS")
do_command("*RST")

# =====
# Capture:
# =====
def capture():

    # Use auto-scale to automatically set up oscilloscope.
    print "Autoscale."
    do_command(":AUToscale")

    # Set trigger mode.
    do_command(":TRIGger:MODE EDGE")
    qresult = do_query_string(":TRIGger:MODE?")
    print "Trigger mode: %s" % qresult

    # Set EDGE trigger parameters.
    do_command(":TRIGger:EDGE:SOURCe CHANnel1")
    qresult = do_query_string(":TRIGger:EDGE:SOURce?")
    print "Trigger edge source: %s" % qresult

    do_command(":TRIGger:EDGE:LEVel 1.5")
    qresult = do_query_string(":TRIGger:EDGE:LEVel?")
    print "Trigger edge level: %s" % qresult

    do_command(":TRIGger:EDGE:SLOPe POSitive")
    qresult = do_query_string(":TRIGger:EDGE:SLOPe?")
    print "Trigger edge slope: %s" % qresult

    # Save oscilloscope setup.
    sSetup = do_query_string(":SYSTem:SETup?")
    sSetup = get_definite_length_block_data(sSetup)

    f = open("setup.stp", "wb")
    f.write(sSetup)
    f.close()
    print "Setup bytes saved: %d" % len(sSetup)

    # Change oscilloscope settings with individual commands:

    # Set vertical scale and offset.
    do_command(":CHANnel1:SCALe 0.05")
    qresult = do_query_values(":CHANnel1:SCALe?") [0]
    print "Channel 1 vertical scale: %f" % qresult

    do_command(":CHANnel1:OFFSet -1.5")
    qresult = do_query_values(":CHANnel1:OFFSet?") [0]
    print "Channel 1 offset: %f" % qresult

    # Set horizontal scale and offset.
    do_command(":TIMEbase:SCALe 0.0002")

```

```

qresult = do_query_string(":TIMEbase:SCALe?")
print "Timebase scale: %s" % qresult

do_command(":TIMEbase:POSIon 0.0")
qresult = do_query_string(":TIMEbase:POSIon?")
print "Timebase position: %s" % qresult

# Set the acquisition type.
do_command(":ACQuire:TYPE NORMal")
qresult = do_query_string(":ACQuire:TYPE?")
print "Acquire type: %s" % qresult

# Or, set up oscilloscope by loading a previously saved setup.
sSetup = ""
f = open("setup.stp", "rb")
sSetup = f.read()
f.close()
do_command(":SYSTem:SETup #8%08d%s" % (len(sSetup), sSetup), hide_param
s=True)
print "Setup bytes restored: %d" % len(sSetup)

# Capture an acquisition using :DIGItize.
do_command(":DIGItize CHANnel1")

# =====
# Analyze:
# =====
def analyze():

    # Make measurements.
    # -----
    do_command(":MEASure:SOURce CHANnel1")
    qresult = do_query_string(":MEASure:SOURce?")
    print "Measure source: %s" % qresult

    do_command(":MEASure:FREQuency")
    qresult = do_query_string(":MEASure:FREQuency?")
    print "Measured frequency on channel 1: %s" % qresult

    do_command(":MEASure:VAMPplitude")
    qresult = do_query_string(":MEASure:VAMPplitude?")
    print "Measured vertical amplitude on channel 1: %s" % qresult

    # Download the screen image.
    # -----
    do_command(":HARDcopy:INKSaver OFF")

    sDisplay = do_query_string(":DISPlay:DATA? PNG, COLOR")
    sDisplay = get_definite_length_block_data(sDisplay)

    # Save display data values to file.
    f = open("screen_image.png", "wb")
    f.write(sDisplay)
    f.close()
    print "Screen image written to screen_image.png."

```

```

# Download waveform data.
# -----
#
# Set the waveform points mode.
do_command(":WAVEform:POINTS:MODE RAW")
qresult = do_query_string(":WAVEform:POINTS:MODE?")
print "Waveform points mode: %s" % qresult

# Get the number of waveform points available.
do_command(":WAVEform:POINTS 10240")
qresult = do_query_string(":WAVEform:POINTS?")
print "Waveform points available: %s" % qresult

# Set the waveform source.
do_command(":WAVEform:SOURce CHANnel1")
qresult = do_query_string(":WAVEform:SOURce?")
print "Waveform source: %s" % qresult

# Choose the format of the data returned:
do_command(":WAVEform:FORMAT BYTE")
print "Waveform format: %s" % do_query_string(":WAVEform:FORMAT?")

# Display the waveform settings from preamble:
wav_form_dict = {
    0 : "BYTE",
    1 : "WORD",
    4 : "ASCII",
}
acq_type_dict = {
    0 : "NORMal",
    1 : "PEAK",
    2 : "AVERage",
    3 : "HRESolution",
}

preamble_string = do_query_string(":WAVEform:PREamble?")
(
    wav_form, acq_type, wfmpnts, avgcnt, x_increment, x_origin,
    x_reference, y_increment, y_origin, y_reference
) = string.split(preamble_string, ",")

print "Waveform format: %s" % wav_form_dict[int(wav_form)]
print "Acquire type: %s" % acq_type_dict[int(acq_type)]
print "Waveform points desired: %s" % wfmpnts
print "Waveform average count: %s" % avgcnt
print "Waveform X increment: %s" % x_increment
print "Waveform X origin: %s" % x_origin
print "Waveform X reference: %s" % x_reference # Always 0.
print "Waveform Y increment: %s" % y_increment
print "Waveform Y origin: %s" % y_origin
print "Waveform Y reference: %s" % y_reference

# Get numeric values for later calculations.
x_increment = do_query_values(":WAVEform:XINCrement?") [0]
x_origin = do_query_values(":WAVEform:XORigin?") [0]
y_increment = do_query_values(":WAVEform:YINCrement?") [0]
y_origin = do_query_values(":WAVEform:YORigin?") [0]

```

```

y_reference = do_query_values(":WAVEform:YREFerence?") [0]

# Get the waveform data.
sData = do_query_string(":WAVEform:DATA?")
sData = get_definite_length_block_data(sData)

# Unpack unsigned byte data.
values = struct.unpack("%dB" % len(sData), sData)
print "Number of data values: %d" % len(values)

# Save waveform data values to CSV file.
f = open("waveform_data.csv", "w")

for i in xrange(0, len(values) - 1):
    time_val = x_origin + (i * x_increment)
    voltage = ((values[i] - y_reference) * y_increment) + y_origin
    f.write("%E, %f\n" % (time_val, voltage))

f.close()
print "Waveform format BYTE data written to waveform_data.csv."


# =====
# Send a command and check for errors:
# =====
def do_command(command, hide_params=False):

    if hide_params:
        (header, data) = string.split(command, " ", 1)
        if debug:
            print "\nCmd = '%s'" % header
    else:
        if debug:
            print "\nCmd = '%s'" % command

    InfiniiVision.write("%s\n" % command)

    if hide_params:
        check_instrument_errors(header)
    else:
        check_instrument_errors(command)


# =====
# Send a query, check for errors, return string:
# =====
def do_query_string(query):
    if debug:
        print "Qys = '%s'" % query
    result = InfiniiVision.ask("%s\n" % query)
    check_instrument_errors(query)
    return result


# =====
# Send a query, check for errors, return values:
# =====

```

```

def do_query_values(query):
    if debug:
        print "Qyv = '%s'" % query
    results = InfiniiVision.ask_for_values("%s\n" % query)
    check_instrument_errors(query)
    return results

# =====
# Check for instrument errors:
# =====
def check_instrument_errors(command):

    while True:
        error_string = InfiniiVision.ask(":SYSTem:ERRor?\n")
        if error_string:    # If there is an error string value.

            if error_string.find("+0,", 0, 3) == -1:    # Not "No error".

                print "ERROR: %s, command: '%s'" % (error_string, command)
                print "Exited because of error."
                sys.exit(1)

            else:    # "No error"
                break

        else:    # :SYSTem:ERRor? should always return string.
            print "ERROR: :SYSTem:ERRor? returned nothing, command: '%s'" % command
            print "Exited because of error."
            sys.exit(1)

# =====
# Returns data from definite-length block.
# =====
def get_definite_length_block_data(sBlock):

    # First character should be "#".
    pound = sBlock[0:1]
    if pound != "#":
        print "PROBLEM: Invalid binary block format, pound char is '%s'." % pound
        print "Exited because of problem."
        sys.exit(1)

    # Second character is number of following digits for length value.
    digits = sBlock[1:2]

    # Get the data out of the block and return it.
    sData = sBlock[int(digits) + 2:]

    return sData

# =====
# Main program:

```

```

# =====
InfiniiVision = visa.instrument("TCPIP0::130.29.70.139::inst0::INSTR")
InfiniiVision.timeout = 15
InfiniiVision.term_chars = ""
InfiniiVision.clear()

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

print "End of program."

```

## VISA Example in Python (PyVISA 1.6 and newer)

You can use the Python programming language with the PyVISA package to control Keysight Infinium Series oscilloscopes.

The Python language and PyVISA package can be downloaded from the web at <http://www.python.org/> and <http://pyvisa.readthedocs.org/>, respectively.

To run this example with Python and PyVISA:

- 1** Cut-and-paste the code that follows into a file named "example.py".
- 2** Edit the program to use the VISA address of your oscilloscope.
- 3** If "python.exe" can be found via your PATH environment variable, open a Command Prompt window; then, change to the folder that contains the "example.py" file, and enter:

```

python example.py

# *****
# This program illustrates a few commonly-used programming
# features of your Keysight oscilloscope.
# *****

# Import modules.
# -----
import visa
import string
import struct
import sys

# Global variables (booleans: 0 = False, 1 = True).
# -----
debug = 0

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.

```

```

idn_string = do_query_string("*IDN?")
print "Identification string: '%s'" % idn_string

# Clear status and load the default setup.
do_command("*CLS")
do_command("*RST")

# =====
# Capture:
# =====
def capture():

    # Use auto-scale to automatically set up oscilloscope.
    print "Autoscale."
    do_command(":AUToscale")

    # Set trigger mode.
    do_command(":TRIGger:MODE EDGE")
    qresult = do_query_string(":TRIGger:MODE?")
    print "Trigger mode: %s" % qresult

    # Set EDGE trigger parameters.
    do_command(":TRIGger:EDGE:SOURCe CHANnel1")
    qresult = do_query_string(":TRIGger:EDGE:SOURCe?")
    print "Trigger edge source: %s" % qresult

    do_command(":TRIGger:EDGE:LEVel 1.5")
    qresult = do_query_string(":TRIGger:EDGE:LEVel?")
    print "Trigger edge level: %s" % qresult

    do_command(":TRIGger:EDGE:SLOPe POSitive")
    qresult = do_query_string(":TRIGger:EDGE:SLOPe?")
    print "Trigger edge slope: %s" % qresult

    # Save oscilloscope setup.
    sSetup = do_query_ieee_block(":SYSTem:SETup?")

    f = open("setup.stp", "wb")
    f.write(sSetup)
    f.close()
    print "Setup bytes saved: %d" % len(sSetup)

    # Change oscilloscope settings with individual commands:

    # Set vertical scale and offset.
    do_command(":CHANnel1:SCALe 0.05")
    qresult = do_query_string(":CHANnel1:SCALe?")
    print "Channel 1 vertical scale: %s" % qresult

    do_command(":CHANnel1:OFFSet -1.5")
    qresult = do_query_string(":CHANnel1:OFFSet?")
    print "Channel 1 offset: %s" % qresult

    # Set horizontal scale and offset.
    do_command(":TIMEbase:SCALe 0.0002")
    qresult = do_query_string(":TIMEbase:SCALe?")

```

```

print "Timebase scale: %s" % qresult

do_command(":TIMEbase:POSIon 0.0")
qresult = do_query_string(":TIMEbase:POSITION?")
print "Timebase position: %s" % qresult

# Set the acquisition type.
do_command(":ACQuire:TYPE NORMAL")
qresult = do_query_string(":ACQuire:TYPE?")
print "Acquire type: %s" % qresult

# Or, set up oscilloscope by loading a previously saved setup.
sSetup = ""
f = open("setup.stp", "rb")
sSetup = f.read()
f.close()
do_command_ieee_block(":SYSTem:SETup", sSetup)
print "Setup bytes restored: %d" % len(sSetup)

# Capture an acquisition using :DIGitize.
do_command(":DIGitize CHANnel1")

# =====
# Analyze:
# -----
def analyze():

    # Make measurements.
    # -----
    do_command(":MEASure:SOURce CHANNEL1")
    qresult = do_query_string(":MEASure:SOURce?")
    print "Measure source: %s" % qresult

    do_command(":MEASure:FREQuency")
    qresult = do_query_string(":MEASure:FREQuency?")
    print "Measured frequency on channel 1: %s" % qresult

    do_command(":MEASure:VAMPplitude")
    qresult = do_query_string(":MEASure:VAMPplitude?")
    print "Measured vertical amplitude on channel 1: %s" % qresult

    # Download the screen image.
    # -----
    do_command(":HARDcopy:INKSaver OFF")

    sDisplay = do_query_ieee_block(":DISPlay:DATA? PNG, COLOR")

    # Save display data values to file.
    f = open("screen_image.png", "wb")
    f.write(sDisplay)
    f.close()
    print "Screen image written to screen_image.png."

    # Download waveform data.
    # -----

```

```

# Set the waveform points mode.
do_command(":WAVeform:POINts:MODE RAW")
qresult = do_query_string(":WAVeform:POINts:MODE?")
print "Waveform points mode: %s" % qresult

# Get the number of waveform points available.
do_command(":WAVeform:POINts 10240")
qresult = do_query_string(":WAVeform:POINts?")
print "Waveform points available: %s" % qresult

# Set the waveform source.
do_command(":WAVeform:SOURce CHANnel1")
qresult = do_query_string(":WAVeform:SOURce?")
print "Waveform source: %s" % qresult

# Choose the format of the data returned:
do_command(":WAVeform:FORMAT BYTE")
print "Waveform format: %s" % do_query_string(":WAVeform:FORMAT?")

# Display the waveform settings from preamble:
wav_form_dict = {
    0 : "BYTE",
    1 : "WORD",
    4 : "ASCII",
}
acq_type_dict = {
    0 : "NORMal",
    1 : "PEAK",
    2 : "AVERage",
    3 : "HRESolution",
}

preamble_string = do_query_string(":WAVeform:PREamble?")
(
    wav_form, acq_type, wfmpnts, avgcnt, x_increment, x_origin,
    x_reference, y_increment, y_origin, y_reference
) = string.split(preamble_string, ",")

print "Waveform format: %s" % wav_form_dict[int(wav_form)]
print "Acquire type: %s" % acq_type_dict[int(acq_type)]
print "Waveform points desired: %s" % wfmpnts
print "Waveform average count: %s" % avgcnt
print "Waveform X increment: %s" % x_increment
print "Waveform X origin: %s" % x_origin
print "Waveform X reference: %s" % x_reference # Always 0.
print "Waveform Y increment: %s" % y_increment
print "Waveform Y origin: %s" % y_origin
print "Waveform Y reference: %s" % y_reference

# Get numeric values for later calculations.
x_increment = do_query_number(":WAVeform:XINCrement?")
x_origin = do_query_number(":WAVeform:XORigin?")
y_increment = do_query_number(":WAVeform:YINCrement?")
y_origin = do_query_number(":WAVeform:YORigin?")
y_reference = do_query_number(":WAVeform:YREFerence?")

# Get the waveform data.

```

```

sData = do_query_ieee_block(":WAVEform:DATA?")

# Unpack unsigned byte data.
values = struct.unpack("%dB" % len(sData), sData)
print "Number of data values: %d" % len(values)

# Save waveform data values to CSV file.
f = open("waveform_data.csv", "w")

for i in xrange(0, len(values) - 1):
    time_val = x_origin + (i * x_increment)
    voltage = ((values[i] - y_reference) * y_increment) + y_origin
    f.write("%E, %f\n" % (time_val, voltage))

f.close()
print "Waveform format BYTE data written to waveform_data.csv."


# =====
# Send a command and check for errors:
# =====
def do_command(command, hide_params=False):

    if hide_params:
        (header, data) = string.split(command, " ", 1)
        if debug:
            print "\nCmd = '%s'" % header
        else:
            if debug:
                print "\nCmd = '%s'" % command

    InfiniiVision.write("%s" % command)

    if hide_params:
        check_instrument_errors(header)
    else:
        check_instrument_errors(command)


# =====
# Send a command and binary values and check for errors:
# =====
def do_command_ieee_block(command, values):
    if debug:
        print "Cmb = '%s'" % command
    InfiniiVision.write_binary_values("%s" % command, values, datatype='c')
    check_instrument_errors(command)


# =====
# Send a query, check for errors, return string:
# =====
def do_query_string(query):
    if debug:
        print "Qys = '%s'" % query
    result = InfiniiVision.query("%s" % query)

```

```

check_instrument_errors(query)
return result

# =====
# Send a query, check for errors, return floating-point value:
# =====
def do_query_number(query):
    if debug:
        print "Qyn = '%s'" % query
    results = InfiniiVision.query("%s" % query)
    check_instrument_errors(query)
    return float(results)

# =====
# Send a query, check for errors, return binary values:
# =====
def do_query_ieee_block(query):
    if debug:
        print "Qys = '%s'" % query
    result = InfiniiVision.query_binary_values("%s" % query, datatype='s')
    check_instrument_errors(query)
    return result[0]

# =====
# Check for instrument errors:
# =====
def check_instrument_errors(command):

    while True:
        error_string = InfiniiVision.query(":SYSTem:ERRor?")
        if error_string:    # If there is an error string value.

            if error_string.find("+0," , 0, 3) == -1:    # Not "No error".

                print "ERROR: %s, command: '%s'" % (error_string, command)
                print "Exited because of error."
                sys.exit(1)

            else:    # "No error"
                break

        else:    # :SYSTem:ERRor? should always return string.
            print "ERROR: :SYSTem:ERRor? returned nothing, command: '%s'" % command
            print "Exited because of error."
            sys.exit(1)

# =====
# Main program:
# =====

rm = visa.ResourceManager()
InfiniiVision= rm.open_resource("TCPIP0::130.29.70.139::inst0::INSTR")

```

```
InfiniiVision.timeout = 15000
InfiniiVision.clear()

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

print "End of program."
```

## SICL Examples

- "SICL Example in C" on page 1430
- "SICL Example in Visual Basic" on page 1439

### SICL Example in C

To compile and run this example in Microsoft Visual Studio 2008:

- 1 Open Visual Studio.
- 2 Create a new Visual C++, Win32, Win32 Console Application project.
- 3 In the Win32 Application Wizard, click **Next >**. Then, check **Empty project**, and click **Finish**.
- 4 Cut-and-paste the code that follows into a file named "example.c" in the project directory.
- 5 In Visual Studio 2008, right-click the Source Files folder, choose **Add > Add Existing Item...**, select the example.c file, and click **Add**.
- 6 Edit the program to use the SICL address of your oscilloscope.
- 7 Choose **Project > Properties....** In the Property Pages dialog, update these project settings:
  - a Under Configuration Properties, Linker, Input, add "sicl32.lib" to the Additional Dependencies field.
  - b Under Configuration Properties, C/C++, Code Generation, select Multi-threaded DLL for the Runtime Library field.
  - c Click **OK** to close the Property Pages dialog.
- 8 Add the include files and library files search paths:
  - a Choose **Tools > Options....**
  - b In the Options dialog, select **VC++ Directories** under Projects and Solutions.
  - c Show directories for **Include files**, and add the include directory (for example, Program Files\Keysight\IO Libraries Suite\include).
  - d Show directories for **Library files**, and add the library files directory (for example, Program Files\Keysight\IO Libraries Suite\lib).
  - e Click **OK** to close the Options dialog.
- 9 Build and run the program.

```
/*
 * Keysight SICL Example in C
 * -----
 * This program illustrates a few commonly-used programming
 * features of your Keysight oscilloscope.
 */

#include <stdio.h>           /* For printf() . */
```

```

#include <string.h>           /* For strcpy(), strcat(). */
#include <time.h>            /* For clock(). */
#include <sicl.h>             /* Keysight SICL routines. */

#define SICL_ADDRESS      "usb0[2391::6054::US50210029::0]"
#define TIMEOUT          5000
#define IEEEBLOCK_SPACE  300000

/* Function prototypes */
void initialize(void);           /* Initialize to known state. */
void capture(void);              /* Capture the waveform. */
void analyze(void);              /* Analyze the captured waveform. */

void do_command(char *command);   /* Send command. */
int do_command_ieeeblock(char *command); /* Command w/IEEE block. */
void do_query_string(char *query); /* Query for string. */
void do_query_number(char *query); /* Query for number. */
void do_query_numbers(char *query); /* Query for numbers. */
int do_query_ieeeblock(char *query); /* Query for IEEE block. */
void check_instrument_errors();   /* Check for inst errors. */

/* Global variables */
INST id;                         /* Device session ID. */
char str_result[256] = {0};        /* Result from do_query_string(). */
double num_result;                /* Result from do_query_number(). */
unsigned char ieeeblock_data[IEEEBLOCK_SPACE]; /* Result from
   do_query_ieeeblock(). */
double dbl_results[10];           /* Result from do_query_numbers(). */

/* Main Program
 * -----
void main(void)
{
    /* Install a default SICL error handler that logs an error message
     * and exits. On Windows 98SE or Windows Me, view messages with
     * the SICL Message Viewer. For Windows 2000 or XP, use the Event
     * Viewer.
    */
    ionerror(I_ERROR_EXIT);

    /* Open a device session using the SICL_ADDRESS */
    id = iopen(SICL_ADDRESS);

    if (id == 0)
    {
        printf ("Oscilloscope iopen failed!\n");
    }
    else
    {
        printf ("Oscilloscope session opened!\n");
    }

    /* Initialize - start from a known state. */
    initialize();

    /* Capture data. */
    capture();
}

```

```

/* Analyze the captured waveform. */
analyze();

/* Close the device session to the instrument. */
iclose(id);
printf ("Program execution is complete...\n");

/* For WIN16 programs, call _siclcleanup before exiting to release
 * resources allocated by SICL for this application. This call is
 * a no-op for WIN32 programs.
 */
_siclcleanup();
}

/* Initialize the oscilloscope to a known state.
 * -----
void initialize (void)
{
    /* Set the I/O timeout value for this session to 5 seconds. */
    itimeout(id, TIMEOUT);

    /* Clear the interface. */
    iclear(id);

    /* Get and display the device's *IDN? string. */
    do_query_string("*IDN?");
    printf("Oscilloscope *IDN? string: %s\n", str_result);

    /* Clear status and load the default setup. */
    do_command("*CLS");
    do_command("*RST");
}

/* Capture the waveform.
 * -----
void capture (void)
{
    int num_bytes;
    FILE *fp;

    /* Use auto-scale to automatically configure oscilloscope.
     * -----
    do_command(":AUToscale");

    /* Set trigger mode (EDGE, PULSe, PATTern, etc., and input source. */
    do_command(":TRIGger:MODE EDGE");
    do_query_string(":TRIGger:MODE?");
    printf("Trigger mode: %s\n", str_result);

    /* Set EDGE trigger parameters. */
    do_command(":TRIGger:EDGE:SOURCe CHANnel1");
    do_query_string(":TRIGger:EDGE:SOURCe?");
    printf("Trigger edge source: %s\n", str_result);

    do_command(":TRIGger:EDGE:LEVel 1.5");
    do_query_string(":TRIGger:EDGE:LEVel?");
}

```

```

printf("Trigger edge level: %s\n", str_result);

do_command(":TRIGger:EDGE:SLOPe POSitive");
do_query_string(":TRIGger:EDGE:SLOPe?");
printf("Trigger edge slope: %s\n", str_result);

/* Save oscilloscope configuration.
 * -----
 */

/* Read system setup. */
num_bytes = do_query_ieeeblock(":SYSTem:SETUp?");
printf("Read setup string query (%d bytes).\n", num_bytes);

/* Write setup string to file. */
fp = fopen ("c:\\scope\\config\\setup.stp", "wb");
num_bytes = fwrite(ieeeblock_data, sizeof(unsigned char), num_bytes,
    fp);
fclose (fp);
printf("Wrote setup string (%d bytes) to ", num_bytes);
printf("c:\\scope\\config\\setup.stp.\n");

/* Change settings with individual commands:
 * -----
 */

/* Set vertical scale and offset. */
do_command(":CHANnel1:SCALe 0.05");
do_query_string(":CHANnel1:SCALe?");
printf("Channel 1 vertical scale: %s\n", str_result);

do_command(":CHANnel1:OFFSet -1.5");
do_query_string(":CHANnel1:OFFSet?");
printf("Channel 1 offset: %s\n", str_result);

/* Set horizontal scale and position. */
do_command(":TIMEbase:SCALe 0.0002");
do_query_string(":TIMEbase:SCALe?");
printf("Timebase scale: %s\n", str_result);

do_command(":TIMEbase:POSIon 0.0");
do_query_string(":TIMEbase:POSIon?");
printf("Timebase position: %s\n", str_result);

/* Set the acquisition type (NORMAL, PEAK, AVERAGE, or HRESolution). */
do_command(":ACQuire:TYPE NORMAL");
do_query_string(":ACQuire:TYPE?");
printf("Acquire type: %s\n", str_result);

/* Or, configure by loading a previously saved setup.
 * -----
 */

/* Read setup string from file. */
fp = fopen ("c:\\scope\\config\\setup.stp", "rb");
num_bytes = fread (ieeeblock_data, sizeof(unsigned char),
    IEEEBLOCK_SPACE, fp);
fclose (fp);
printf("Read setup string (%d bytes) from file ", num_bytes);

```

```

printf("c:\\\\scope\\\\config\\\\setup.stp.\n");

/* Restore setup string. */
num_bytes = do_command_ieeeblock(":SYSTem:SETUp", num_bytes);
printf("Restored setup string (%d bytes).\n", num_bytes);

/* Capture an acquisition using :DIGitize.
 * -----
do_command(":DIGitize CHANnel1");
}

/* Analyze the captured waveform.
 * -----
void analyze (void)
{
    double wav_format;
    double acq_type;
    double wav_points;
    double avg_count;
    double x_increment;
    double x_origin;
    double x_reference;
    double y_increment;
    double y_origin;
    double y_reference;

    FILE *fp;
    int num_bytes; /* Number of bytes returned from instrument. */
    int i;

    /* Make a couple of measurements.
     * -----
do_command(":MEASure:SOURce CHANnel1");
do_query_string(":MEASure:SOURce?");
printf("Measure source: %s\n", str_result);

do_command(":MEASure:FREQuency");
do_query_number(":MEASure:FREQuency?");
printf("Frequency: %.4f kHz\n", num_result / 1000);

do_command(":MEASure:VAMPplitude");
do_query_number(":MEASure:VAMPplitude?");
printf("Vertical amplitude: %.2f V\n", num_result);

/* Download the screen image.
 * -----
do_command(":HARDcopy:INKSaver OFF");

/* Read screen image. */
num_bytes = do_query_ieeeblock(":DISPlay:DATA? PNG, COLOR");
printf("Screen image bytes: %d\n", num_bytes);

/* Write screen image bytes to file. */
fp = fopen ("c:\\\\scope\\\\data\\\\screen.png", "wb");
num_bytes = fwrite(ieeeblock_data, sizeof(unsigned char), num_bytes,
                  fp);
fclose (fp);
}

```

```

printf("Wrote screen image (%d bytes) to ", num_bytes);
printf("c:\\scope\\data\\screen.png.\n");

/* Download waveform data.
 * -----
 */

/* Set the waveform points mode. */
do_command(":WAVeform:POINTs:MODE RAW");
do_query_string(":WAVeform:POINTs:MODE?");
printf("Waveform points mode: %s\n", str_result);

/* Get the number of waveform points available. */
do_command(":WAVeform:POINTs 10240");
do_query_string(":WAVeform:POINTs?");
printf("Waveform points available: %s\n", str_result);

/* Set the waveform source. */
do_command(":WAVeform:SOURce CHANnel1");
do_query_string(":WAVeform:SOURce?");
printf("Waveform source: %s\n", str_result);

/* Choose the format of the data returned (WORD, BYTE, ASCII): */
do_command(":WAVeform:FORMAT BYTE");
do_query_string(":WAVeform:FORMAT?");
printf("Waveform format: %s\n", str_result);

/* Display the waveform settings: */
do_query_numbers(":WAVeform:PREamble?");

wav_format = dbl_results[0];
if (wav_format == 0.0)
{
    printf("Waveform format: BYTE\n");
}
else if (wav_format == 1.0)
{
    printf("Waveform format: WORD\n");
}
else if (wav_format == 2.0)
{
    printf("Waveform format: ASCii\n");
}

acq_type = dbl_results[1];
if (acq_type == 0.0)
{
    printf("Acquire type: NORMAl\n");
}
else if (acq_type == 1.0)
{
    printf("Acquire type: PEAK\n");
}
else if (acq_type == 2.0)
{
    printf("Acquire type: AVERage\n");
}
else if (acq_type == 3.0)
{
    printf("Acquire type: HISTOgram\n");
}

```

```

{
    printf("Acquire type: HRESolution\n");
}

wav_points = dbl_results[2];
printf("Waveform points: %e\n", wav_points);

avg_count = dbl_results[3];
printf("Waveform average count: %e\n", avg_count);

x_increment = dbl_results[4];
printf("Waveform X increment: %e\n", x_increment);

x_origin = dbl_results[5];
printf("Waveform X origin: %e\n", x_origin);

x_reference = dbl_results[6];
printf("Waveform X reference: %e\n", x_reference);

y_increment = dbl_results[7];
printf("Waveform Y increment: %e\n", y_increment);

y_origin = dbl_results[8];
printf("Waveform Y origin: %e\n", y_origin);

y_reference = dbl_results[9];
printf("Waveform Y reference: %e\n", y_reference);

/* Read waveform data. */
num_bytes = do_query_ieeeblock(":WAVEFORM:DATA?");
printf("Number of data values: %d\n", num_bytes);

/* Open file for output. */
fp = fopen("c:\\scope\\data\\waveform_data.csv", "wb");

/* Output waveform data in CSV format. */
for (i = 0; i < num_bytes - 1; i++)
{
    /* Write time value, voltage value. */
    fprintf(fp, "%9f, %6f\n",
            x_origin + ((float)i * x_increment),
            (((float)ieeeblock_data[i] - y_reference) * y_increment)
            + y_origin);
}

/* Close output file. */
fclose(fp);
printf("Waveform format BYTE data written to ");
printf("c:\\scope\\data\\waveform_data.csv.\n");
}

/* Send a command to the instrument.
 * -----
void do_command(command)
char *command;
{
    char message[80];

```

```

    strcpy(message, command);
    strcat(message, "\n");
    sprintf(id, message);

    check_instrument_errors();
}

/* Command with IEEE definite-length block.
 * -----
int do_command_ieeeblock(command, num_bytes)
char *command;
int num_bytes;
{
    char message[80];
    int data_length;

    strcpy(message, command);
    strcat(message, "#8%08d");
    sprintf(id, message, num_bytes);
    ifwrite(id, ieeeblock_data, num_bytes, 1, &data_length);

    check_instrument_errors();

    return(data_length);
}

/* Query for a string result.
 * -----
void do_query_string(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");
    sprintf(id, message);

    iscanf(id, "%t\n", str_result);

    check_instrument_errors();
}

/* Query for a number result.
 * -----
void do_query_number(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");
    sprintf(id, message);

    iscanf(id, "%lf", &num_result);

    check_instrument_errors();
}

```

```

    }

/* Query for numbers result.
 * -----
void do_query_numbers(query)
char *query;
{
    char message[80];

    strcpy(message, query);
    strcat(message, "\n");
    iprintf(id, message);

    iscanf(id, "%,10lf\n", dbl_results);

    check_instrument_errors();
}

/* Query for an IEEE definite-length block result.
 * -----
int do_query_ieeeblock(query)
char *query;
{
    char message[80];
    int data_length;

    strcpy(message, query);
    strcat(message, "\n");
    iprintf(id, message);

    data_length = IEEEBLOCK_SPACE;
    iscanf(id, "%#b", &data_length, ieeeblock_data);

    if (data_length == IEEEBLOCK_SPACE )
    {
        printf("IEEE block buffer full: ");
        printf("May not have received all data.\n");
    }

    check_instrument_errors();

    return(data_length);
}

/* Check for instrument errors.
 * -----
void check_instrument_errors()
{
    char str_err_val[256] = {0};
    char str_out[800] = "";

    ipromptf(id, ":SYSTem:ERRor?\n", "%t", str_err_val);
    while(strncmp(str_err_val, "+0,No error", 3) != 0 )
    {
        strcat(str_out, ", ");
        strcat(str_out, str_err_val);
        ipromptf(id, ":SYSTem:ERRor?\n", "%t", str_err_val);
    }
}

```

```

    }

    if (strcmp(str_out, "") != 0)
    {
        printf("INST Error%s\n", str_out);
        iflush(id, I_BUF_READ | I_BUF_WRITE);
    }
}

```

## SICL Example in Visual Basic

To run this example in Visual Basic for Applications:

- 1 Start the application that provides Visual Basic for Applications (for example, Microsoft Excel).
- 2 Press ALT+F11 to launch the Visual Basic editor.
- 3 Add the sicl32.bas file to your project:
  - a Choose **File > Import File....**
  - b Navigate to the header file, sicl32.bas (installed with Keysight IO Libraries Suite and found in the Program Files\Keysight\IO Libraries Suite\include directory), select it, and click **Open**.
- 4 Choose **Insert > Module**.
- 5 Cut-and-paste the code that follows into the editor.
- 6 Edit the program to use the SICL address of your oscilloscope, and save the changes.
- 7 Run the program.

```

'
' Keysight SICL Example in Visual Basic
' -----
' This program illustrates a few commonly-used programming
' features of your Keysight oscilloscope.
' -----


Option Explicit

Public id As Integer      ' Session to instrument.

' Declare variables to hold numeric values returned by
' ivscanf/ifread.
Public dblQueryResult As Double
Public Const ByteArraySize = 5000000
Public retCount As Long
Public byteArray(ByteArraySize) As Byte

' Declare fixed length string variable to hold string value returned
' by ivscanf.
Public strQueryResult As String * 200

' For Sleep subroutine.

```

```

Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'
' Main Program
' -----
'

Sub Main()

    On Error GoTo ErrorHandler

    ' Open a device session using the SICL_ADDRESS.
    id = iopen("usb0[2391::6054::US50210029::0]")
    Call itimeout(id, 5000)

    ' Initialize - start from a known state.
    Initialize

    ' Capture data.
    Capture

    ' Analyze the captured waveform.
    Analyze

    ' Close the vi session and the resource manager session.
    Call iclose(id)

    Exit Sub

ErrorHandler:

    MsgBox "*** Error : " + Error, vbExclamation
    End

End Sub

'
' Initialize the oscilloscope to a known state.
' -----
'

Private Sub Initialize()

    On Error GoTo ErrorHandler

    ' Clear the interface.
    Call iclear(id)

    ' Get and display the device's *IDN? string.
    strQueryResult = DoQueryString("*IDN?")
    MsgBox "Result is: " + RTrim(strQueryResult), vbOKOnly, "*IDN? Result"

    ' Clear status and load the default setup.
    DoCommand "*CLS"
    DoCommand "*RST"

    Exit Sub

ErrorHandler:

```

```

    MsgBox "*** Error : " + Error, vbExclamation
End

End Sub

'

' Capture the waveform.
' -----
Private Sub Capture()

On Error GoTo ErrorHandler

' Use auto-scale to automatically configure oscilloscope.
' -----
DoCommand ":AUToscale"

' Set trigger mode (EDGE, PULSe, PATTern, etc., and input source.
DoCommand ":TRIGger:MODE EDGE"
Debug.Print "Trigger mode: " + _
DoQueryString(":TRIGger:MODE?")

' Set EDGE trigger parameters.
DoCommand ":TRIGger:EDGE:SOURCe CHANnel1"
Debug.Print "Trigger edge source: " + _
DoQueryString(":TRIGger:EDGE:SOURce?")

DoCommand ":TRIGger:EDGE:LEVel 1.5"
Debug.Print "Trigger edge level: " + _
DoQueryString(":TRIGger:EDGE:LEVel?")

DoCommand ":TRIGger:EDGE:SLOPe POSitive"
Debug.Print "Trigger edge slope: " + _
DoQueryString(":TRIGger:EDGE:SLOPe?")

' Save oscilloscope configuration.
' -----
Dim lngSetupStringSize As Long
lngSetupStringSize = DoQueryIEEEBlock_Bytes(":SYSTem:SETup?")
Debug.Print "Setup bytes saved: " + CStr(lngSetupStringSize)

' Output setup string to a file:
Dim strPath As String
strPath = "c:\scope\config\setup.dat"
If Len(Dir(strPath)) Then
    Kill strPath      ' Remove file if it exists.
End If

' Open file for output.
Dim hFile As Long
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Dim lngI As Long
For lngI = 0 To lngSetupStringSize - 1
    Put hFile, , byteArray(lngI)      ' Write data.
Next lngI

```

```

        Close hFile      ' Close file.

        ' Change settings with individual commands:
        ' -----
        ' Set vertical scale and offset.
        DoCommand ":CHANnel1:SCALe 0.05"
        Debug.Print "Channel 1 vertical scale: " + _
                    DoQueryString(":CHANnel1:SCALe?")

        DoCommand ":CHANnel1:OFFSet -1.5"
        Debug.Print "Channel 1 vertical offset: " + _
                    DoQueryString(":CHANnel1:OFFSet?")

        ' Set horizontal scale and position.
        DoCommand ":TIMEbase:SCALe 0.0002"
        Debug.Print "Timebase scale: " + _
                    DoQueryString(":TIMEbase:SCALe?")

        DoCommand ":TIMEbase:POSIon 0.0"
        Debug.Print "Timebase position: " + _
                    DoQueryString(":TIMEbase:POSIon?")

        ' Set the acquisition type (NORMal, PEAK, AVERage, or HRESolution).
        DoCommand ":ACQuire:TYPE NORMal"
        Debug.Print "Acquire type: " + _
                    DoQueryString(":ACQuire:TYPE?")

        ' Or, configure by loading a previously saved setup.
        ' -----
        strPath = "c:\scope\config\setup.dat"
        Open strPath For Binary Access Read As hFile      ' Open file for input.
        Dim lngSetupFileSize As Long
        lngSetupFileSize = LOF(hFile)      ' Length of file.
        Get hFile, , byteArray      ' Read data.
        Close hFile      ' Close file.
        ' Write setup string back to oscilloscope using ":SYSTem:SETup"
        ' command:
        Dim lngRestored As Long
        lngRestored = DoCommandIEEEBlock(":SYSTem:SETup", lngSetupFileSize)
        Debug.Print "Setup bytes restored: " + CStr(lngRestored)

        ' Capture an acquisition using :DIGitize.
        ' -----
        DoCommand ":DIGitize CHANnel1"

        Exit Sub

ErrorHandler:
        MsgBox "*** Error : " + Error, vbExclamation
        End

End Sub

        '
        ' Analyze the captured waveform.

```

```

' -----
Private Sub Analyze()

On Error GoTo ErrorHandler

' Make a couple of measurements.
' -----
DoCommand ":MEASure:SOURce CHANnel1"
Debug.Print "Measure source: " + _
DoQueryString(":MEASure:SOURce?")

DoCommand ":MEASure:FREQuency"
dblQueryResult = DoQueryNumber(":MEASure:FREQuency?")
MsgBox "Frequency:" + vbCrLf + _
FormatNumber(dblQueryResult / 1000, 4) + " kHz"

DoCommand ":MEASure:VAMPplitude"
dblQueryResult = DoQueryNumber(":MEASure:VAMPplitude?")
MsgBox "Vertical amplitude:" + vbCrLf + _
FormatNumber(dblQueryResult, 4) + " V"

' Download the screen image.
' -----
DoCommand ":HARDcopy:INKSaver OFF"

' Get screen image.
Dim lngBlockSize As Long
lngBlockSize = DoQueryIEEEBlock_Bytes(":DISPLAY:DATA? PNG, COLOR")
Debug.Print "Screen image bytes: " + CStr(lngBlockSize)

' Save screen image to a file:
Dim strPath As String
strPath = "c:\scope\data\screen.png"
If Len(Dir(strPath)) Then
    Kill strPath      ' Remove file if it exists.
End If
Dim hFile As Long
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Dim lngI As Long
' Skip past header.
For lngI = CInt(Chr(byteArray(1))) + 2 To lngBlockSize - 1
    Put hFile, , byteArray(lngI)      ' Write data.
Next lngI
Close hFile      ' Close file.
MsgBox "Screen image written to " + strPath

' Download waveform data.
' -----
' Set the waveform points mode.
DoCommand ":WAVEform:POINTS:MODE RAW"
Debug.Print "Waveform points mode: " + _
DoQueryString(":WAVEform:POINTS:MODE?")

' Get the number of waveform points available.

```

```

DoCommand ":WAVeform:POINTs 10240"
Debug.Print "Waveform points available: " + _
    DoQueryString(":WAVeform:POINTs?")

' Set the waveform source.
DoCommand ":WAVeform:SOURce CHANnel1"
Debug.Print "Waveform source: " + _
    DoQueryString(":WAVeform:SOURce?")

' Choose the format of the data returned (WORD, BYTE, ASCII):
DoCommand ":WAVeform:FORMAT BYTE"
Debug.Print "Waveform format: " + _
    DoQueryString(":WAVeform:FORMAT?")

' Display the waveform settings:
Dim Preamble() As Double
Dim intFormat As Integer
Dim intType As Integer
Dim lngPoints As Long
Dim lngCount As Long
Dim dblXIncrement As Double
Dim dblXOrigin As Double
Dim lngXReference As Long
Dim sngYIncrement As Single
Dim sngYOrigin As Single
Dim lngYReference As Long

Preamble() = DoQueryNumbers(":WAVeform:PREamble?")

intFormat = Preamble(0)
intType = Preamble(1)
lngPoints = Preamble(2)
lngCount = Preamble(3)
dblXIncrement = Preamble(4)
dblXOrigin = Preamble(5)
lngXReference = Preamble(6)
sngYIncrement = Preamble(7)
sngYOrigin = Preamble(8)
lngYReference = Preamble(9)

If intFormat = 0 Then
    Debug.Print "Waveform format: BYTE"
ElseIf intFormat = 1 Then
    Debug.Print "Waveform format: WORD"
ElseIf intFormat = 2 Then
    Debug.Print "Waveform format: ASCII"
End If

If intType = 0 Then
    Debug.Print "Acquisition type: NORMAL"
ElseIf intType = 1 Then
    Debug.Print "Acquisition type: PEAK"
ElseIf intType = 2 Then
    Debug.Print "Acquisition type: AVERAGE"
ElseIf intType = 3 Then
    Debug.Print "Acquisition type: HRESolution"
End If

```

```

Debug.Print "Waveform points: " + _
FormatNumber(lngPoints, 0)

Debug.Print "Waveform average count: " + _
FormatNumber(lngCount, 0)

Debug.Print "Waveform X increment: " + _
Format(dblXIncrement, "Scientific")

Debug.Print "Waveform X origin: " + _
Format(dblXOrigin, "Scientific")

Debug.Print "Waveform X reference: " + _
FormatNumber(lngXReference, 0)

Debug.Print "Waveform Y increment: " + _
Format(sngYIncrement, "Scientific")

Debug.Print "Waveform Y origin: " + _
FormatNumber(sngYOrigin, 0)

Debug.Print "Waveform Y reference: " + _
FormatNumber(lngYReference, 0)

' Get the waveform data
Dim lngNumBytes As Long
lngNumBytes = DoQueryIEEEBlock_Bytes(":WAVEFORM:DATA?")
Debug.Print "Number of data values: " + _
CStr(lngNumBytes - CInt(Chr(byteArray(1))) - 2)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"

' Open file for output.
Open strPath For Output Access Write Lock Write As hFile

' Output waveform data in CSV format.
Dim lngDataValue As Long

' Skip past header.
For lngI = CInt(Chr(byteArray(1))) + 2 To lngNumBytes - 2
    lngDataValue = CLng(byteArray(lngI))

    ' Write time value, voltage value.
    Print #hFile, _
        FormatNumber(dblXOrigin + (lngI * dblXIncrement), 9) + _
        ", " + _
        FormatNumber(((lngDataValue - lngYReference) * _
        sngYIncrement) + sngYOrigin)

Next lngI

' Close output file.
Close hFile      ' Close file.
MsgBox "Waveform format BYTE data written to " + _
"c:\scope\data\waveform_data.csv."

```

```

        Exit Sub

ErrorHandler:
    MsgBox "*** Error : " + Error, vbExclamation
End

End Sub

Private Sub DoCommand(command As String)

On Error GoTo ErrorHandler

Call ivprintf(id, command + vbCrLf)

CheckInstrumentErrors

Exit Sub

ErrorHandler:
    MsgBox "*** Error : " + Error, vbExclamation
End

End Sub

Private Function DoCommandIEEEBlock(command As String, _
    lngBlockSize As Long)

On Error GoTo ErrorHandler

' Send command part.
Call ivprintf(id, command + " ")

' Write definite-length block bytes.
Call ifwrite(id, byteArray(), lngBlockSize, vbNull, retCount)

' retCount is now actual number of bytes written.
DoCommandIEEEBlock = retCount

CheckInstrumentErrors

Exit Function

ErrorHandler:
    MsgBox "*** Error : " + Error, vbExclamation
End

End Function

Private Function DoQueryString(query As String) As String

Dim actual As Long

On Error GoTo ErrorHandler

```

```

Dim strResult As String * 200

Call ivprintf(id, query + vbLf)
Call ivscanf(id, "%200t", strResult)
DoQueryString = strResult

CheckInstrumentErrors

Exit Function

ErrorHandler:

MsgBox "*** Error : " + Error, vbExclamation
End

End Function

Private Function DoQueryNumber(query As String) As Double

On Error GoTo ErrorHandler

Dim dblResult As Double

Call ivprintf(id, query + vbLf)
Call ivscanf(id, "%lf" + vbLf, dblResult)
DoQueryNumber = dblResult

CheckInstrumentErrors

Exit Function

ErrorHandler:

MsgBox "*** Error : " + Error, vbExclamation
End

End Function

Private Function DoQueryNumbers(query As String) As Double()

On Error GoTo ErrorHandler

Dim dblResults(10) As Double

Call ivprintf(id, query + vbLf)
Call ivscanf(id, "%,10lf" + vbLf, dblResults)
DoQueryNumbers = dblResults

CheckInstrumentErrors

Exit Function

ErrorHandler:

MsgBox "*** Error : " + Error, vbExclamation
End

```

```

End Function

Private Function DoQueryIEEEBlock_Bytes(query As String) As Long

On Error GoTo ErrorHandler

' Send query.
Call ivprintf(id, query + vbLf)

' Read definite-length block bytes.
Sleep 2000      ' Delay before reading data.
Call ifread(id, byteArray(), ByteArraySize, vbNull, retCount)

' Get number of block length digits.
Dim intLengthDigits As Integer
intLengthDigits = CInt(Chr(byteArray(1)))

' Get block length from those digits.
Dim strBlockLength As String
strBlockLength = ""
Dim i As Integer
For i = 2 To intLengthDigits + 1
    strBlockLength = strBlockLength + Chr(byteArray(i))
Next

' Return number of bytes in block plus header.
DoQueryIEEEBlock_Bytes = CLng(strBlockLength) + intLengthDigits + 2

CheckInstrumentErrors

Exit Function

ErrorHandler:

MsgBox "*** Error : " + Error, vbExclamation
End

End Function

Private Sub CheckInstrumentErrors()

On Error GoTo ErrorHandler

Dim strErrVal As String * 200
Dim strOut As String

Call ivprintf(id, ":SYSTem:ERRor?" + vbLf)      ' Query any errors data.
Call ivscanf(id, "%200t", strErrVal)      ' Read: Errnum, "Error String".
While Val(strErrVal) <> 0                  ' End if find: +0,"No Error".
    strOut = strOut + "INST Error: " + strErrVal
    Call ivprintf(id, ":SYSTem:ERRor?" + vbLf)      ' Request error message
    Call ivscanf(id, "%200t", strErrVal)      ' Read error message.
Wend

If Not strOut = "" Then

```

```
    MsgBox strOut, vbExclamation, "INST Error Messages"
    Call iflush(id, I_BUF_READ Or I_BUF_WRITE)

End If

Exit Sub

ErrorHandler:

    MsgBox "*** Error : " + Error, vbExclamation
    End

End Sub
```

## SCPI.NET Examples

These programming examples show how to use the SCPI.NET drivers that come with Keysight's free Command Expert software.

While you can write code manually using SCPI.NET drivers (as described in this section), you can also use the Command Expert software to:

- Connect to instruments and control them interactively using SCPI command sets.
- Quickly prototype and test command sequences.
- Generate C#, VB.NET, or C/C++ code for command sequences.
- Find, download, and install SCPI command sets.
- Browse command trees, search for commands, and view command descriptions.

The Command Expert suite also comes with Add-ons for easy instrument control and measurement data retrieval in NI LabVIEW, Microsoft Excel, Keysight VEE, and Keysight SystemVue.

For more information on Keysight Command Expert, and to download the software, see: <http://www.keysight.com/find/commandexpert>

- ["SCPI.NET Example in C#" on page 1450](#)
- ["SCPI.NET Example in Visual Basic .NET" on page 1456](#)
- ["SCPI.NET Example in IronPython" on page 1462](#)

### SCPI.NET Example in C#

To compile and run this example in Microsoft Visual Studio 2008:

- 1 Install the Keysight Command Expert software and the command set for the oscilloscope.
- 2 Open Visual Studio.
- 3 Create a new Visual C#, Windows, Console Application project.
- 4 Cut-and-paste the code that follows into the C# source file.
- 5 Edit the program to use the address of your oscilloscope.
- 6 Add a reference to the SCPI.NET driver:
  - a Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
  - b Choose **Add Reference....**
  - c In the Add Reference dialog, select the **Browse** tab, and navigate to the ScpiNetDrivers folder.

- Windows XP: C:\Documents and Settings\All Users\Keysight\Command Expert\ScpiNetDrivers
  - Windows 7: C:\ProgramData\Keysight\Command Expert\ScpiNetDrivers
- d Select the .dll file for your oscilloscope, for example **AgInfiniiVision3000X\_02\_00.dll**; then, click **OK**.

**7** Build and run the program.

For more information, see the SCPI.NET driver help that comes with Keysight Command Expert.

```
/*
 * Keysight SCPI.NET Example in C#
 * -----
 * This program illustrates a few commonly used programming
 * features of your Keysight oscilloscope.
 * -----
 */

using System;
using System.IO;
using System.Text;
using Keysight.CommandExpert.ScpiNet.AgInfiniiVision3000X_02_00;

namespace InfiniiVision
{
    class ScpiNetInstrumentApp
    {
        private static AgInfiniiVision3000X myScope;

        static void Main(string[] args)
        {
            try
            {
                string strScopeAddress;
                //strScopeAddress = "a-mx3054a-60028.cos.keysight.com";
                strScopeAddress =
                    "TCPIPO::a-mx4054a-60154.cos.keysight.com::inst0::INSTR";
                Console.WriteLine("Connecting to oscilloscope...");
                Console.WriteLine();
                myScope = new AgInfiniiVision3000X(strScopeAddress);
                myScope.Transport.DefaultTimeout.Set(10000);

                // Initialize - start from a known state.
                Initialize();

                // Capture data.
                Capture();

                // Analyze the captured waveform.
                Analyze();

                Console.WriteLine("Press any key to exit");
                Console.ReadKey();
            }
        }
    }
}
```

```

        catch (System.ApplicationException err)
        {
            Console.WriteLine("*** SCPI.NET Error : " + err.Message);
        }
        catch (System.SystemException err)
        {
            Console.WriteLine("*** System Error Message : " + err.Message);
        }
        catch (System.Exception err)
        {
            System.Diagnostics.Debug.Fail("Unexpected Error");
            Console.WriteLine("*** Unexpected Error : " + err.Message);
        }
    finally
    {
        //myScope.Dispose();
    }

}

/*
 * Initialize the oscilloscope to a known state.
 * -----
 */
private static void Initialize()
{
    string strResults;

    // Get and display the device's *IDN? string.
    myScope.SCPI.IDN.Query(out strResults);
    Console.WriteLine("*IDN? result is: {0}", strResults);

    // Clear status and load the default setup.
    myScope.SCPI.CLS.Command();
    myScope.SCPI.RST.Command();
}

/*
 * Capture the waveform.
 * -----
 */
private static void Capture()
{
    string strResults;
    double fResult;

    // Use auto-scale to automatically configure oscilloscope.
    myScope.SCPI.AUToscale.Command(null, null, null, null, null);

    // Set trigger mode.
    myScope.SCPI.TRIGger.MODE.Command("EDGE");
    myScope.SCPI.TRIGger.MODE.Query(out strResults);
    Console.WriteLine("Trigger mode: {0}", strResults);

    // Set EDGE trigger parameters.
    myScope.SCPI.TRIGger.EDGE.SOURce.Command("CHANnel1");
    myScope.SCPI.TRIGger.EDGE.SOURce.Query(out strResults);
}

```

```

Console.WriteLine("Trigger edge source: {0}", strResults);

myScope.SCPI.TRIGger.EDGE.LEVEL.Command(1.5, "CHANnel1");
myScope.SCPI.TRIGger.EDGE.LEVEL.Query("CHANnel1", out fResult);
Console.WriteLine("Trigger edge level: {0:F2}", fResult);

myScope.SCPI.TRIGger.EDGE.SLOPe.Command("POSitive");
myScope.SCPI.TRIGger.EDGE.SLOPe.Query(out strResults);
Console.WriteLine("Trigger edge slope: {0}", strResults);

// Save oscilloscope configuration.
string[] strResultsArray; // Results array.
int nLength; // Number of bytes returned from instrument.
string strPath;

// Query and read setup string.
myScope.SCPI.SYSTem.SETup.Query(out strResultsArray);
nLength = strResultsArray.Length;

// Write setup string to file.
strPath = "c:\\scope\\config\\setup.stp";
File.WriteAllLines(strPath, strResultsArray);
Console.WriteLine("Setup bytes saved: {0}", nLength);

// Change settings with individual commands:

// Set vertical scale and offset.
myScope.SCPI.CHANnel.SCALE.Command(1, 0.05);
myScope.SCPI.CHANnel.SCALE.Query(1, out fResult);
Console.WriteLine("Channel 1 vertical scale: {0:F4}", fResult);

myScope.SCPI.CHANnel.OFFSet.Command(1, -1.5);
myScope.SCPI.CHANnel.OFFSet.Query(1, out fResult);
Console.WriteLine("Channel 1 vertical offset: {0:F4}", fResult);

// Set horizontal scale and offset.
myScope.SCPI.TIMebase.SCALE.Command(0.0002);
myScope.SCPI.TIMebase.SCALE.Query(out fResult);
Console.WriteLine("Timebase scale: {0:F4}", fResult);

myScope.SCPI.TIMebase.POSition.Command(0.0);
myScope.SCPI.TIMebase.POSition.Query(out fResult);
Console.WriteLine("Timebase position: {0:F2}", fResult);

// Set the acquisition type.
myScope.SCPI.ACQuire.TYPE.Command("NORMAL");
myScope.SCPI.ACQuire.TYPE.Query(out strResults);
Console.WriteLine("Acquire type: {0}", strResults);

// Or, configure by loading a previously saved setup.
int nBytesWritten;

strPath = "c:\\scope\\config\\setup.stp";
strResultsArray = File.ReadAllLines(strPath);
nBytesWritten = strResultsArray.Length;

```

```

    // Restore setup string.
    myScope.SCPI.SYSTem.SETup.Command(strResultsArray);
    Console.WriteLine("Setup bytes restored: {0}", nBytesWritten);

    // Capture an acquisition using :DIGItize.
    myScope.SCPI.DIGItize.Command("CHANnel1", null, null, null, null);
}

/*
 * Analyze the captured waveform.
 * -----
 */
private static void Analyze()
{
    string strResults, source1, source2;
    double fResult;

    // Make a couple of measurements.
    // -----
    myScope.SCPI.MEASure.SOURce.Command("CHANnel1", null);
    myScope.SCPI.MEASure.SOURce.Query(out source1, out source2);
    Console.WriteLine("Measure source: {0}", source1);

    myScope.SCPI.MEASure.FREQuency.Command("CHANnel1");
    myScope.SCPI.MEASure.FREQuency.Query("CHANnel1", out fResult);
    Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000);

    // Use direct command/query when commands not in command set.
    myScope.Transport.Command.Invoke(":MEASure:VAMplitude CHANnel1");
    myScope.Transport.Query.Invoke(":MEASure:VAMplitude? CHANnel1",
        out strResults);
    Console.WriteLine("Vertical amplitude: {0} V", strResults);

    // Download the screen image.
    // -----
    myScope.SCPI.HARDcopy.INKSaver.Command(false);

    // Get the screen data.
    byte[] byteResultsArray;    // Results array.
    myScope.SCPI.DISPlay.DATA.Query("PNG", "COLOR",
        out byteResultsArray);
    int nLength;    // Number of bytes returned from instrument.
    nLength = byteResultsArray.Length;

    // Store the screen data to a file.
    string strPath;
    strPath = "c:\\scope\\data\\screen.png";
    FileStream fStream = File.Open(strPath, FileMode.Create);
    fStream.Write(byteResultsArray, 0, nLength);
    fStream.Close();
    Console.WriteLine("Screen image ({0} bytes) written to {1}",
        nLength, strPath);

    // Download waveform data.
    // -----
}

// Set the waveform points mode.

```

```

myScope.SCPI.WAVEform.POINTs.MODE.Command("RAW");
myScope.SCPI.WAVEform.POINTs.MODE.Query(out strResults);
Console.WriteLine("Waveform points mode: {0}", strResults);

// Get the number of waveform points available.
myScope.SCPI.WAVEform.POINTs.CommandPoints(10240);
int nPointsAvail;
myScope.SCPI.WAVEform.POINTs.Query1(out nPointsAvail);
Console.WriteLine("Waveform points available: {0}", nPointsAvail);

// Set the waveform source.
myScope.SCPI.WAVEform.SOURce.Command("CHANnel1");
myScope.SCPI.WAVEform.SOURce.Query(out strResults);
Console.WriteLine("Waveform source: {0}", strResults);

// Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.SCPI.WAVEform.FORMat.Command("BYTE");
myScope.SCPI.WAVEform.FORMat.Query(out strResults);
Console.WriteLine("Waveform format: {0}", strResults);

// Display the waveform settings:
int nFormat, nType, nPoints, nCount, nXreference, nYreference;
double dblXincrement, dblXorigin, dblYincrement, dblYorigin;
myScope.SCPI.WAVEform.PREamble.Query(
    out nFormat,
    out nType,
    out nPoints,
    out nCount,
    out dblXincrement,
    out dblXorigin,
    out nXreference,
    out dblYincrement,
    out dblYorigin,
    out nYreference);

if (nFormat == 0)
{
    Console.WriteLine("Waveform format: BYTE");
}
else if (nFormat == 1)
{
    Console.WriteLine("Waveform format: WORD");
}
else if (nFormat == 2)
{
    Console.WriteLine("Waveform format: ASCII");
}

if (nType == 0)
{
    Console.WriteLine("Acquire type: NORMAL");
}
else if (nType == 1)
{
    Console.WriteLine("Acquire type: PEAK");
}
else if (nType == 2)

```

```

{
    Console.WriteLine("Acquire type: AVERage");
}
else if (nType == 3)
{
    Console.WriteLine("Acquire type: HRESolution");
}

Console.WriteLine("Waveform points: {0:e}", nPoints);
Console.WriteLine("Waveform average count: {0:e}", nCount);
Console.WriteLine("Waveform X increment: {0:e}", dblXincrement);
Console.WriteLine("Waveform X origin: {0:e}", dblXorigin);
Console.WriteLine("Waveform X reference: {0:e}", nXreference);
Console.WriteLine("Waveform Y increment: {0:e}", dblYincrement);
Console.WriteLine("Waveform Y origin: {0:e}", dblYorigin);
Console.WriteLine("Waveform Y reference: {0:e}", nYreference);

// Read waveform data.
myScope.SCPI.WAVeform.DATA.QueryBYTE(out byteResultsArray);
nLength = byteResultsArray.Length;
Console.WriteLine("Number of data values: {0}", nLength);

// Set up output file:
strPath = "c:\\scope\\data\\waveform_data.csv";
if (File.Exists(strPath)) File.Delete(strPath);

// Open file for output.
StreamWriter writer = File.CreateText(strPath);

// Output waveform data in CSV format.
for (int i = 0; i < nLength - 1; i++)
    writer.WriteLine("{0:f9}, {1:f6}",
        dblXorigin + ((float)i * dblXincrement),
        (((float)byteResultsArray[i] - nYreference)
         * dblYincrement) + dblYorigin);

// Close output file.
writer.Close();
Console.WriteLine("Waveform format BYTE data written to {0}",
    strPath);
}
}

```

SCPI.NET Example in Visual Basic .NET

To compile and run this example in Microsoft Visual Studio 2008:

- 1 Install the Keysight Command Expert software and the command set for the oscilloscope.
  - 2 Open Visual Studio.
  - 3 Create a new Visual Basic, Windows, Console Application project.
  - 4 Cut-and-paste the code that follows into the Visual Basic .NET source file.

- 5** Edit the program to use the VISA address of your oscilloscope.
- 6** Add a reference to the SCPI.NET 3.0 driver:
  - a** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
  - b** Choose **Add Reference...**.
  - c** In the Add Reference dialog, select the **Browse** tab, and navigate to the ScpiNetDrivers folder.
    - Windows XP: C:\Documents and Settings\All Users\Keysight\Command Expert\ScpiNetDrivers
    - Windows 7: C:\ProgramData\Keysight\Command Expert\ScpiNetDrivers
  - d** Select the .dll file for your oscilloscope, for example **AgInfiniiVision3000X\_02\_00.dll**; then, click **OK**.
  - e** Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment and choose **Properties**; then, select "InfiniiVision.ScpiNetInstrumentApp" as the **Startup object**.
- 7** Build and run the program.

For more information, see the SCPI.NET driver help that comes with Keysight Command Expert.

```
'-----'
' Keysight SCPI.NET Example in Visual Basic .NET
' -----
' This program illustrates a few commonly used programming
' features of your Keysight oscilloscope.
' -----'

Imports System
Imports System.IO
Imports System.Text
Imports Keysight.CommandExpert.ScpiNet.AgInfiniiVision3000X_02_00

Namespace InfiniiVision
    Class ScpiNetInstrumentApp
        Private Shared myScope As AgInfiniiVision3000X

        Public Shared Sub Main(ByVal args As String())
            Try
                Dim strScopeAddress As String
                'strScopeAddress = "a-mx3054a-60028.cos.keysight.com";
                strScopeAddress =
                    "TCPIP0::a-mx4054a-60154.cos.keysight.com::inst0::INSTR"
                Console.WriteLine("Connecting to oscilloscope...")
                Console.WriteLine()
                myScope = New AgInfiniiVision3000X(strScopeAddress)
                myScope.Transport.DefaultTimeout.[Set](10000)

                ' Initialize - start from a known state.
                Initialize()
            End Try
        End Sub
    End Class
End Namespace
```

```

' Capture data.
Capture()

' Analyze the captured waveform.
Analyze()

Console.WriteLine("Press any key to exit")
Console.ReadKey()
Catch err As System.ApplicationException
    Console.WriteLine("**** SCPI.NET Error : " & err.Message)
Catch err As System.SystemException
    Console.WriteLine("**** System Error Message : " & err.Message)
Catch err As System.Exception
    System.Diagnostics.Debug.Fail("Unexpected Error")
    Console.WriteLine("**** Unexpected Error : " & err.Message)
    'myScope.Dispose();
Finally
End Try

End Sub

' Initialize the oscilloscope to a known state.
' -----
Private Shared Sub Initialize()
    Dim strResults As String

    ' Get and display the device's *IDN? string.
    myScope.SCPI.IDN.Query(strResults)
    Console.WriteLine("*IDN? result is: {0}", strResults)

    ' Clear status and load the default setup.
    myScope.SCPI.CLS.Command()
    myScope.SCPI.RST.Command()
End Sub

' Capture the waveform.
' -----
Private Shared Sub Capture()
    Dim strResults As String
    Dim fResult As Double

    ' Use auto-scale to automatically configure oscilloscope.
    myScope.SCPI.AUToscale.Command(Nothing, Nothing, Nothing, _
        Nothing, Nothing)

    ' Set trigger mode.
    myScope.SCPI.TRIGger.MODE.Command("EDGE")
    myScope.SCPI.TRIGger.MODE.Query(strResults)
    Console.WriteLine("Trigger mode: {0}", strResults)

    ' Set EDGE trigger parameters.
    myScope.SCPI.TRIGger.EDGE.SOURce.Command("CHANnel1")
    myScope.SCPI.TRIGger.EDGE.SOURce.Query(strResults)
    Console.WriteLine("Trigger edge source: {0}", strResults)

```

```

myScope.SCPI.TRIGger.EDGE.LEVel.Command(1.5, "CHANnel1")
myScope.SCPI.TRIGger.EDGE.LEVel.Query("CHANnel1", fResult)
Console.WriteLine("Trigger edge level: {0:F2}", fResult)

myScope.SCPI.TRIGger.EDGE.SLOPe.Command("POSitive")
myScope.SCPI.TRIGger.EDGE.SLOPe.Query(strResults)
Console.WriteLine("Trigger edge slope: {0}", strResults)

' Save oscilloscope configuration.
Dim strResultsArray As String()
' Results array.
Dim nLength As Integer
' Number of bytes returned from instrument.
Dim strPath As String

' Query and read setup string.
myScope.SCPI.SYSTem.SETup.Query(strResultsArray)
nLength = strResultsArray.Length

' Write setup string to file.
strPath = "c:\scope\config\setup.stp"
File.WriteAllLines(strPath, strResultsArray)
Console.WriteLine("Setup bytes saved: {0}", nLength)

' Change settings with individual commands:

' Set vertical scale and offset.
myScope.SCPI.CHANnel.SCALE.Command(1, 0.05)
myScope.SCPI.CHANnel.SCALE.Query(1, fResult)
Console.WriteLine("Channel 1 vertical scale: {0:F4}", fResult)

myScope.SCPI.CHANnel.OFFSet.Command(1, -1.5)
myScope.SCPI.CHANnel.OFFSet.Query(1, fResult)
Console.WriteLine("Channel 1 vertical offset: {0:F4}", fResult)

' Set horizontal scale and offset.
myScope.SCPI.TIMebase.SCALE.Command(0.0002)
myScope.SCPI.TIMebase.SCALE.Query(fResult)
Console.WriteLine("Timebase scale: {0:F4}", fResult)

myScope.SCPI.TIMebase.POSition.Command(0.0)
myScope.SCPI.TIMebase.POSition.Query(fResult)
Console.WriteLine("Timebase position: {0:F2}", fResult)

' Set the acquisition type.
myScope.SCPI.ACQuire.TYPE.Command("NORMAL")
myScope.SCPI.ACQuire.TYPE.Query(strResults)
Console.WriteLine("Acquire type: {0}", strResults)

' Or, configure by loading a previously saved setup.
Dim nBytesWritten As Integer

strPath = "c:\scope\config\setup.stp"
strResultsArray = File.ReadAllLines(strPath)
nBytesWritten = strResultsArray.Length

```

```

' Restore setup string.
myScope.SCPI.SYSTem.SETup.Command(strResultsArray)
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten)

' Capture an acquisition using :DIGItize.
myScope.SCPI.DIGItize.Command("CHANnel1", Nothing, Nothing, _
                               Nothing, Nothing)
End Sub

' Analyze the captured waveform.
' -----
Private Shared Sub Analyze()
    Dim strResults As String, source1 As String, source2 As String
    Dim fResult As Double

    ' Make a couple of measurements.
    ' -----
    myScope.SCPI.MEASure.SOURce.Command("CHANnel1", Nothing)
    myScope.SCPI.MEASure.SOURce.Query(source1, source2)
    Console.WriteLine("Measure source: {0}", source1)

    myScope.SCPI.MEASure.FREQuency.Command("CHANnel1")
    myScope.SCPI.MEASure.FREQuency.Query("CHANnel1", fResult)
    Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000)

    ' Use direct command/query when commands not in command set.
    myScope.Transport.Command.Invoke(":MEASure:VAMplitude CHANnel1")
    myScope.Transport.Query.Invoke(":MEASure:VAMplitude? CHANnel1", _
                                    strResults)
    Console.WriteLine("Vertical amplitude: {0} V", strResults)

    ' Download the screen image.
    ' -----
    myScope.SCPI.HARDcopy.INKSaver.Command(False)

    ' Get the screen data.
    Dim byteResultsArray As Byte()
    ' Results array.
    myScope.SCPI.DISPlay.DATA.Query("PNG", "COLOR", byteResultsArray)
    Dim nLength As Integer
    ' Number of bytes returned from instrument.
    nLength = byteResultsArray.Length

    ' Store the screen data to a file.
    Dim strPath As String
    strPath = "c:\scope\data\screen.png"
    Dim fStream As FileStream = File.Open(strPath, FileMode.Create)
    fStream.Write(byteResultsArray, 0, nLength)
    fStream.Close()
    Console.WriteLine("Screen image ({0} bytes) written to {1}", _
                     nLength, strPath)

    ' Download waveform data.
    ' -----

```

```

' Set the waveform points mode.
myScope.SCPI.WAVEform.POINts.MODE.Command("RAW")
myScope.SCPI.WAVEform.POINts.MODE.Query(strResults)
Console.WriteLine("Waveform points mode: {0}", strResults)

' Get the number of waveform points available.
myScope.SCPI.WAVEform.POINts.CommandPoints(10240)
Dim nPointsAvail As Integer
myScope.SCPI.WAVEform.POINts.Query1(nPointsAvail)
Console.WriteLine("Waveform points available: {0}", nPointsAvail)

' Set the waveform source.
myScope.SCPI.WAVEform.SOURce.Command("CHANnel1")
myScope.SCPI.WAVEform.SOURce.Query(strResults)
Console.WriteLine("Waveform source: {0}", strResults)

' Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.SCPI.WAVEform.FORMat.Command("BYTE")
myScope.SCPI.WAVEform.FORMat.Query(strResults)
Console.WriteLine("Waveform format: {0}", strResults)

' Display the waveform settings:
Dim nFormat As Integer, nType As Integer, nPoints As Integer, _
    nCount As Integer, nXreference As Integer, _
    nYreference As Integer
Dim dblXincrement As Double, dblXorigin As Double, _
    dblYincrement As Double, dblYorigin As Double
myScope.SCPI.WAVEform.PREAMble.Query(nFormat, nType, nPoints, _
    nCount, dblXincrement, dblXorigin, nXreference, _
    dblYincrement, dblYorigin, nYreference)

If nFormat = 0 Then
    Console.WriteLine("Waveform format: BYTE")
ElseIf nFormat = 1 Then
    Console.WriteLine("Waveform format: WORD")
ElseIf nFormat = 2 Then
    Console.WriteLine("Waveform format: ASCII")
End If

If nType = 0 Then
    Console.WriteLine("Acquire type: NORMAL")
ElseIf nType = 1 Then
    Console.WriteLine("Acquire type: PEAK")
ElseIf nType = 2 Then
    Console.WriteLine("Acquire type: AVERAGE")
ElseIf nType = 3 Then
    Console.WriteLine("Acquire type: HRESolution")
End If

Console.WriteLine("Waveform points: {0:e}", nPoints)
Console.WriteLine("Waveform average count: {0:e}", nCount)
Console.WriteLine("Waveform X increment: {0:e}", dblXincrement)
Console.WriteLine("Waveform X origin: {0:e}", dblXorigin)
Console.WriteLine("Waveform X reference: {0:e}", nXreference)
Console.WriteLine("Waveform Y increment: {0:e}", dblYincrement)
Console.WriteLine("Waveform Y origin: {0:e}", dblYorigin)
Console.WriteLine("Waveform Y reference: {0:e}", nYreference)

```

```

' Read waveform data.
myScope.SCPI.WAVEform.DATA.QueryBYTE(byteResultsArray)
nLength = byteResultsArray.Length
Console.WriteLine("Number of data values: {0}", nLength)

' Set up output file:
strPath = "c:\scope\data\waveform_data.csv"
If File.Exists(strPath) Then
    File.Delete(strPath)
End If

' Open file for output.
Dim writer As StreamWriter = File.CreateText(strPath)

' Output waveform data in CSV format.
For i As Integer = 0 To nLength - 2
    writer.WriteLine("{0:f9}, {1:f6}", _
        dblXorigin + (CSng(i) * dblXincrement), _
        ((CSng(byteResultsArray(i)) - nYreference) * _
        dblYincrement) + dblYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format BYTE data written to {0}", _
    strPath)
End Sub
End Class
End Namespace

```

## SCPI.NET Example in IronPython

You can also control Keysight oscilloscopes using the SCPI.NET library and Python programming language on the .NET platform using:

- IronPython (<http://ironpython.codeplex.com/>) which is an implementation of the Python programming language running under .NET.

To run this example with IronPython:

- 1 Install the Keysight Command Expert software and the command set for the oscilloscope.
- 2 Cut-and-paste the code that follows into a file named "example.py".
- 3 Edit the program to use the address of your oscilloscope.
- 4 If the IronPython "ipy.exe" can be found via your PATH environment variable, open a Command Prompt window; then, change to the folder that contains the "example.py" file, and enter:

```

ipy example.py

#
# Keysight SCPI.NET Example in IronPython
# ****

```

```

# This program illustrates a few commonly used programming
# features of your Keysight oscilloscope.
# ****
#
# Import Python modules.
# -----
import sys
sys.path.append("C:\Python26\Lib")    # Python Standard Library.
sys.path.append("C:\ProgramData\Keysight\Command Expert\ScpiNetDrivers")
import string

# Import .NET modules.
# -----
from System import *
from System.IO import *
from System.Text import *
from System.Runtime.InteropServices import *
import clr
clr.AddReference("AgInfiniiVision4000X_01_20")
from Keysight.CommandExpert.ScpiNet.AgInfiniiVision4000X_01_20 import *

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.
    idn_string = scope.SCPI.IDN.Query()
    print "Identification string '%s'" % idn_string

    # Clear status and load the default setup.
    scope.SCPI.CLS.Command()
    scope.SCPI.RST.Command()

# =====
# Capture:
# =====
def capture():

    # Use auto-scale to automatically set up oscilloscope.
    print "Autoscale."
    scope.SCPI.AUToscale.Command(None, None, None, None, None)

    # Set trigger mode.
    scope.SCPI.TRIGger.MODE.Command("EDGE")
    qresult = scope.SCPI.TRIGger.MODE.Query()
    print "Trigger mode: %s" % qresult

    # Set EDGE trigger parameters.
    scope.SCPI.TRIGger.EDGE.SOURce.Command("CHANnel1")
    qresult = scope.SCPI.TRIGger.EDGE.SOURce.Query()
    print "Trigger edge source: %s" % qresult

    scope.SCPI.TRIGger.EDGE.LEVel.Command(1.5, "CHANnel1")
    qresult = scope.SCPI.TRIGger.EDGE.LEVel.Query("CHANnel1")

```

```

        print "Trigger edge level: %s" % qresult

        scope.SCPI.TRIGger.EDGE.SLOPe.Command("POSitive")
        qresult = scope.SCPI.TRIGger.EDGE.SLOPe.Query()
        print "Trigger edge slope: %s" % qresult

        # Save oscilloscope setup.
        setup_lines = scope.SCPI.SYSTem.SETup.Query()
        nLength = len(setup_lines)
        File.WriteAllLines("setup.stp", setup_lines)
        print "Setup lines saved: %d" % nLength

        # Change oscilloscope settings with individual commands:

        # Set vertical scale and offset.
        scope.SCPI.CHANnel.SCALE.Command(1, 0.05)
        qresult = scope.SCPI.CHANnel.SCALE.Query(1)
        print "Channel 1 vertical scale: %f" % qresult

        scope.SCPI.CHANnel.OFFSet.Command(1, -1.5)
        qresult = scope.SCPI.CHANnel.OFFSet.Query(1)
        print "Channel 1 offset: %f" % qresult

        # Set horizontal scale and offset.
        scope.SCPI.TIMebase.SCALE.Command(0.0002)
        qresult = scope.SCPI.TIMebase.SCALE.Query()
        print "Timebase scale: %f" % qresult

        scope.SCPI.TIMebase.POSition.Command(0.0)
        qresult = scope.SCPI.TIMebase.POSition.Query()
        print "Timebase position: %f" % qresult

        # Set the acquisition type.
        scope.SCPI.ACQuire.TYPE.Command("NORMAL")
        qresult = scope.SCPI.ACQuire.TYPE.Query()
        print "Acquire type: %s" % qresult

        # Or, set up oscilloscope by loading a previously saved setup.
        setup_lines = File.ReadAllLines("setup.stp")
        scope.SCPI.SYSTem.SETup.Command(setup_lines)
        print "Setup lines restored: %d" % len(setup_lines)

        # Capture an acquisition using :DIGitize.
        scope.SCPI.DIGitize.Command("CHANnel1", None, None, None, None)

        # =====
        # Analyze:
        # =====
        def analyze():

            # Make measurements.
            # -----
            scope.SCPI.MEASure.SOURce.Command("CHANnel1", None)
            (source1, source2) = scope.SCPI.MEASure.SOURce.Query()
            print "Measure source: %s" % source1

```

```

scope.SCPI.MEASure.FREQuency.Command("CHANnel1")
qresult = scope.SCPI.MEASure.FREQuency.Query("CHANnel1")
print "Measured frequency on channel 1: %f" % qresult

# Use direct command/query when commands not in command set.
scope.Transport.Command.Invoke(":MEASure:VAMplitude CHANnel1")
qresult = scope.Transport.Query.Invoke(":MEASure:VAMplitude? CHANnel1")
print "Measured vertical amplitude on channel 1: %s" % qresult

# Download the screen image.
# -----
scope.SCPI.HARDcopy.INKSaver.Command(False)

image_bytes = scope.SCPI.DISPlay.DATA.Query("PNG", "COLOR")
nLength = len(image_bytes)
fStream = File.Open("screen_image.png", FileMode.Create)
fStream.Write(image_bytes, 0, nLength)
fStream.Close()
print "Screen image written to screen_image.png."

# Download waveform data.
# ----

# Set the waveform points mode.
scope.SCPI.WAVEform.POINTs.MODE.Command("RAW")
qresult = scope.SCPI.WAVEform.POINTs.MODE.Query()
print "Waveform points mode: %s" % qresult

# Get the number of waveform points available.
scope.SCPI.WAVEform.POINTs.CommandPoints(10240)
qresult = scope.SCPI.WAVEform.POINTs.Query1()
print "Waveform points available: %s" % qresult

# Set the waveform source.
scope.SCPI.WAVEform.SOURce.Command("CHANnel1")
qresult = scope.SCPI.WAVEform.SOURce.Query()
print "Waveform source: %s" % qresult

# Choose the format of the data returned:
scope.SCPI.WAVEform.FORMAT.Command("BYTE")
qresult = scope.SCPI.WAVEform.FORMAT.Query()
print "Waveform format: %s" % qresult

# Display the waveform settings from preamble:
wav_form_dict = {
    0 : "BYTE",
    1 : "WORD",
    4 : "ASCII",
}
acq_type_dict = {
    0 : "NORMAL",
    1 : "PEAK",
    2 : "AVERage",
    3 : "HRESolution",
}
(

```

```

wav_form, acq_type, wfmpnts, avgcnt, x_increment, x_origin,
x_reference, y_increment, y_origin, y_reference
) = scope.SCPI.WAVEform.PREamble.Query()

print "Waveform format: %s" % wav_form_dict[int(wav_form)]
print "Acquire type: %s" % acq_type_dict[int(acq_type)]
print "Waveform points desired: %s" % wfmpnts
print "Waveform average count: %s" % avgcnt
print "Waveform X increment: %s" % x_increment
print "Waveform X origin: %s" % x_origin
print "Waveform X reference: %s" % x_reference # Always 0.
print "Waveform Y increment: %s" % y_increment
print "Waveform Y origin: %s" % y_origin
print "Waveform Y reference: %s" % y_reference

# Get numeric values for later calculations.
x_increment = scope.SCPI.WAVEform.XINCrement.Query()
x_origin = scope.SCPI.WAVEform.XORigin.Query()
y_increment = scope.SCPI.WAVEform.YINCrement.Query()
y_origin = scope.SCPI.WAVEform.YORigin.Query()
y_reference = scope.SCPI.WAVEform.YREFerence.Query()

# Get the waveform data.
data_bytes = scope.SCPI.WAVEform.DATA.QueryBYTE()
nLength = len(data_bytes)
print "Number of data values: %d" % nLength

# Open file for output.
strPath = "waveform_data.csv"
writer = File.CreateText(strPath)

# Output waveform data in CSV format.
for i in xrange(0, nLength - 1):
    time_val = x_origin + i * x_increment
    voltage = (data_bytes[i] - y_reference) * y_increment + y_origin
    writer.WriteLine("%E, %f" % (time_val, voltage))

# Close output file.
writer.Close()
print "Waveform format BYTE data written to %s." % strPath

# =====
# Main program:
# =====
#addr = "a-mx3054a-60028.cos.keysight.com"
addr = "TCPIP0::a-mx3054a-60028.cos.keysight.com::inst0::INSTR"
scope = AgInfiniiVision4000X(addr)
scope.Transport.DefaultTimeout.Set(10000)

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

print "End of program."

```

```
# Wait for a key press before exiting.  
print "Press any key to exit..."  
Console.ReadKey(True)
```



# Index

## Symbols

+9.9E+37, infinity representation, 1341  
+9.9E+37, measurement error, 477

## Numerics

0 (zero) values in waveform data, 1155  
1 (one) values in waveform data, 1155  
4000 X-Series oscilloscopes, command differences from, 42  
82350B GPIB interface, 6

## A

A429 SEARch commands, 950  
A429 serial bus commands, 728  
absolute value math function, 405  
AC coupling, trigger edge, 1072  
AC input coupling for specified channel, 283  
AC RMS measured on waveform, 495, 536  
accumulate activity, 207  
ACQuire commands, 243  
acquire data, 215, 255  
acquire mode on autoscale, 211  
acquire reset conditions, 192, 1019  
acquire sample rate, 254  
ACQuire subsystem, 61  
acquired data points, 248  
acquisition count, 246  
acquisition mode, 243, 247, 1172  
acquisition type, 243, 255  
acquisition types, 1148  
active edges, 207  
active printer, 420  
activity logic levels, 207  
activity on digital channels, 207  
add function, 1167  
add math function, 405  
add math function as g(t) source, 1250  
address field size, IIC serial decode, 814  
address of network printer, 425  
address, IIC trigger pattern, 817  
Addresses softkey, 50  
AER (Arm Event Register), 208, 227, 229, 1315  
Agilent Flash directory, 690, 708  
alignment, I2S trigger, 796  
all (snapshot) measurement, 478

ALL segments waveform save option, 720  
AM demo signal, 314  
AM depth, waveform generator modulation, 1200  
AM modulation type, waveform generator, 1210  
amplitude, vertical, 491, 529  
amplitude, waveform generator, 1219  
analog channel coupling, 283  
analog channel display, 284  
analog channel impedance, 285  
analog channel input, 1244  
analog channel inversion, 286  
analog channel labels, 287, 342  
analog channel offset, 288  
analog channel protection lock, 1022  
analog channel range, 296  
analog channel scale, 297  
analog channel source for glitch, 1085  
analog channel units, 298  
analog channels only oscilloscopes, 6  
analog probe attenuation, 289  
analog probe head type, 290  
analog probe sensing, 1245  
analog probe skew, 293, 1243  
analysis results, save, 709  
analyzing captured data, 57  
angle brackets, 175  
annotate channels, 287  
annotation background, display, 333  
annotation color, display, 334  
annotation text, display, 335  
annotation X1 position, 336  
annotation Y1 position, 337  
annotation, display, 332  
apparent power, 548  
apply network printer connection settings, 426  
arbitrary waveform generator output, 1196  
arbitrary waveform, byte order, 1185  
arbitrary waveform, capturing from other sources, 1191  
arbitrary waveform, clear, 1188  
arbitrary waveform, download DAC values, 1189  
arbitrary waveform, download floating-point values, 1186  
arbitrary waveform, interpolation, 1190  
arbitrary waveform, points, 1187  
arbitrary waveform, recall, 685  
arbitrary waveform, save, 697  
area for hardcopy print, 419  
area for saved image, 1283

area measurement, 479, 490  
ARINC 429 auto setup, 730  
ARINC 429 base, 731  
ARINC 429 demo signal, 316  
ARINC 429 signal speed, 738  
ARINC 429 signal type, 736  
ARINC 429 source, 737  
ARINC 429 trigger data pattern, 740, 953  
ARINC 429 trigger label, 739, 743, 951  
ARINC 429 trigger SDI pattern, 741, 954  
ARINC 429 trigger SSM pattern, 742, 955  
ARINC 429 trigger type, 744, 952  
ARINC 429 word and error counters, reset, 733  
ARINC 429 word format, 735  
Arm Event Register (AER), 208, 227, 229, 1315  
arm event, NFC arm and event trigger, 1087  
arming edge slope, Edge Then Edge trigger, 1060  
arming edge source, Edge Then Edge trigger, 1061  
arrange waveforms, 1247  
ASCII format, 1157  
ASCII format for data transfer, 1150  
ASCII string, quoted, 175  
ASCIix waveform data format, 717  
assign channel names, 287  
attenuation factor (external trigger) probe, 355  
attenuation for oscilloscope probe, 289  
audio channel, I2S trigger, 805  
AUT option for probe sense, 1245, 1249  
Auto Range capability for DVM, 348  
auto set up, trigger level, 1053  
auto setup (ARINC 429), 730  
auto setup for M1553 trigger, 842  
auto setup for power analysis signals, 651  
auto trigger sweep mode, 1047  
automask create, 573  
automask source, 574  
automask units, 575  
automatic measurements constants, 289  
automatic probe type detection, 1245, 1249  
autoscale, 209  
autoscale acquire mode, 211  
autoscale channels, 212  
AUToscale command, 60  
autoset for FLEXray event trigger, 787  
autoseup for FLEXray decode, 777  
average math function, clear, 390

average value measurement, 492, 530  
 Average, power modulation analysis, 640  
 averaged value math function, 406  
 averaging acquisition type, 244, 1149  
 averaging, synchronizing with, 1330  
 Ax + B math function, 405

**B**

bandwidth filter limits, 354  
 bandwidth filter limits to 20 MHz, 282  
 bar chart of current harmonics results, 625  
 base 10 exponential math function, 405  
 base value measurement, 493, 531  
 base, ARINC 429, 731  
 base, I2S serial decode, 797  
 base, MIL-STD-1553 serial decode, 843  
 base, SENT decode, 853  
 base, UART trigger, 912  
 basic instrument functions, 180  
 baud rate, 758, 828, 901  
 baud rate, CAN FD, 760  
 begin acquisition, 215, 236, 238  
 BHARris window for minimal spectral leakage, 377, 398  
 binary block data, 175, 339, 1030, 1155  
 BINary waveform data format, 717  
 bind levels for masks, 594  
 bit order, 902  
 bit order, SPI decode, 883  
 bit rate measurement, 480  
 bit selection command, bus, 259  
 bit weights, 184  
 bitmap display, 339  
 bits in Service Request Enable Register, 197  
 bits in Standard Event Status Enable Register, 183  
 bits in Status Byte Register, 199  
 bits selection command, bus, 260  
 blank, 214  
 block data, 175, 187, 1030  
 block response data, 64  
 blocking synchronization, 1325  
 blocking wait, 1324  
 BMP format screen image data, 339  
 braces, 174  
 built-in measurements, 57  
 burst data demo signal, 315  
 burst width measurement, 481  
 burst, minimum time before next, 1068  
 bus bit selection command, 259  
 bus bits selection commands, 260  
 bus clear command, 262  
 bus commands, 258  
 BUS data format, 1152  
 bus display, 263  
 bus label command, 264  
 bus mask command, 265  
 BUS<n> commands, 257  
 button disable, 1016

button, calibration protect, 273  
 byte format for data transfer, 1151, 1157  
 BYTeorder, 1153

**C**

C, SICL library example, 1430  
 C, VISA library example, 1377  
 C#, SCPI.NET example, 1450  
 C#, VISA COM example, 1353  
 C#, VISA example, 1396  
 CAL PROTECT button, 273  
 CAL PROTECT switch, 268  
 calculating preshoot of waveform, 509  
 calculating the waveform overshoot, 503  
 calibrate, 269, 270, 273, 277  
 CALibrate commands, 267  
 calibrate date, 269  
 calibrate introduction, 268  
 calibrate label, 270  
 calibrate output, 271  
 calibrate start, 274  
 calibrate status, 275  
 calibrate switch, 273  
 calibrate temperature, 276  
 calibrate time, 277  
 CAN acknowledge, 757  
 CAN and LIN demo signal, 316  
 CAN baud rate, 758  
 CAN demo signal, 315  
 CAN FD baud rate, 760  
 CAN FD data triggers, starting byte position, 769  
 CAN frame counters, reset, 750  
 CAN SEARch commands, 956  
 CAN serial bus commands, 745  
 CAN serial search, data, 959  
 CAN serial search, data length, 960  
 CAN serial search, ID, 961  
 CAN serial search, ID mode, 962  
 CAN serial search, mode, 957  
 CAN signal definition, 759  
 CAN source, 761  
 CAN symbolic data display, 754  
 CAN symbolic data, recall, 686  
 CAN trigger, 762, 768  
 CAN trigger data pattern, 766  
 CAN trigger ID pattern, 770  
 CAN trigger pattern id mode, 771  
 CAN trigger, ID filter for, 765  
 CAN triggering, 723  
 capture data, 215  
 capturing data, 56  
 cardiac waveform generator output, 1195  
 center frequency set, 362, 392, 405  
 center of screen, 1180  
 center reference, 1040  
 center screen, FFT vertical value at, 370, 372  
 center screen, vertical value at, 404, 410  
 CFD demo signal, 317

channel, 242, 287, 1240, 1242  
 channel coupling, 283  
 channel display, 284  
 channel input impedance, 285  
 channel inversion, 286  
 channel label, 287, 1241  
 channel labels, 341, 342  
 channel numbers, 1247  
 channel overload, 295  
 channel protection, 295  
 channel reset conditions, 192, 1019  
 channel selected to produce trigger, 1085, 1126  
 channel signal type, 294  
 channel skew for oscilloscope probe, 293, 1243  
 channel status, 239, 1247  
 channel threshold, 1242  
 channel vernier, 299  
 channel, stop displaying, 214  
 CHANnel<n> commands, 279, 281  
 channels to autoscale, 212  
 channels, how autoscale affects, 209  
 characters to display, 1014  
 chart logic bus state math function, 407  
 chart logic bus state, clock edge, 386  
 chart logic bus state, clock source, 385  
 chart logic bus timing math function, 407  
 chart logic bus, units, 389  
 chart logic bus, value for data = 0, 388  
 chart logic bus, value for data  
     increment, 387  
 classes of input signals, 377, 398  
 classifications, command, 1334  
 clear, 338  
 clear bus command, 262  
 clear cumulative edge variables, 1240  
 clear FFT function, 363  
 clear markers, 482, 1261  
 clear measurement, 482, 1261  
 clear message queue, 181  
 Clear method, 59  
 clear reference waveforms, 1225  
 clear screen, 1248  
 clear status, 181  
 clear waveform area, 330  
 clipped high waveform data value, 1155  
 clipped low waveform data value, 1155  
 clock, 815, 884, 887  
 clock period, SENT signal, 851  
 clock slope, I2S, 798  
 CLOCK source, I2S, 800  
 clock source, setup and hold trigger, 1113  
 clock timeout, SPI, 885  
 clock with infrequent glitch demo  
     signal, 314  
 CLS (Clear Status), 181  
 CME (Command Error) status bit, 183, 185  
 CMOS threshold voltage for digital channels, 327, 1242  
 CMOS trigger threshold voltage, 1288  
 code, :ACQuire:COMplete, 245

code, :ACQuire:SEGmented, 251  
 code, :ACQuire:TYPE, 256  
 code, :AUToscale, 210  
 code, :CHANnel<n>:LABEL, 287  
 code, :CHANnel<n>:PROBe, 289  
 code, :CHANnel<n>:RANGE, 296  
 code, :DIGItize, 216  
 code, :DISPlay:DATA, 339  
 code, :DISPlay:LABEL, 341  
 code, :DISPlay:ORDer, 1247  
 code, :MEASure:PERiod, 518  
 code, :MEASure:RESults, 511  
 code, :MEASure:TDEGe, 526  
 code, :MTEST, 569  
 code, :POD<n>:THReShold, 605  
 code, :RUN:/STOP, 236  
 code, :SYSTem:SETup, 1030  
 code, :TIMEbase:DELay, 1287  
 code, :TIMEbase:MODE, 1037  
 code, :TIMEbase:RANGE, 1039  
 code, :TIMEbase:REFerence, 1040  
 code, :TRIGger:MODE, 1056  
 code, :TRIGger:SLOPe, 1075  
 code, :TRIGger:SOURce, 1076  
 code, :VIEW and :BLANK, 242  
 code, :WAVeform, 1168  
 code, :WAVeform:DATA, 1155  
 code, :WAVeform:POINts, 1159  
 code, :WAVeform:PREamble, 1163  
 code, :WAVeform:SEGmented, 251  
 code, \*RST, 194  
 code, SCPI.NET library example in C#, 1450  
 code, SCPI.NET library example in IronPython, 1462  
 code, SCPI.NET library example in Visual Basic .NET, 1456  
 code, SICL library example in C, 1430  
 code, SICL library example in Visual Basic, 1439  
 code, VISA COM library example in C#, 1353  
 code, VISA COM library example in Python, 1370  
 code, VISA COM library example in Visual Basic, 1344  
 code, VISA COM library example in Visual Basic .NET, 1362  
 code, VISA library example in C, 1377  
 code, VISA library example in C#, 1396  
 code, VISA library example in Python, 1417, 1423  
 code, VISA library example in Visual Basic, 1386  
 code, VISA library example in Visual Basic .NET, 1407  
 colon, root commands prefixed by, 206  
 color palette for hardcopy, 431  
 color palette for image, 703  
 Comma Separated Values (CSV) waveform data format, 717  
 command classifications, 1334  
 command differences from 4000 X-Series oscilloscopes, 42  
 command errors detected in Standard Event Status, 185  
 Command Expert, 1450  
 command header, 1335  
 command headers, common, 1337  
 command headers, compound, 1337  
 command headers, simple, 1337  
 command strings, valid, 1335  
 commands quick reference, 69  
 commands sent over interface, 180  
 commands, more about, 1333  
 commands, obsolete and discontinued, 1233  
 common (\*) commands, 3, 177, 180  
 common command headers, 1337  
 common logarithm math function, 405  
 completion criteria for an acquisition, 245, 246  
 compound command headers, 1337  
 compound header, 1339  
 computer control examples, 1343  
 conditions for external trigger, 353  
 conditions, reset, 192, 1019  
 conduction calculation method for switching loss, 673  
 Config softkey, 50  
 configurations, oscilloscope, 187, 191, 195, 1030  
 Configure softkey, 50  
 connect oscilloscope, 49  
 connect sampled data points, 1246  
 Connection Expert, 51  
 constants for making automatic measurements, 289  
 constants for scaling display factors, 289  
 constants for setting trigger levels, 289  
 control loop response (Bode) power analysis, apply, 613  
 control loop response power analysis, gain/phase view, 616  
 control loop response power analysis, sweep start frequency, 614  
 control loop response power analysis, sweep stop frequency, 615  
 control loop response power analysis, Y maximum, 617  
 control loop response power analysis, Y minimum, 618  
 controller initialization, 56  
 copy display, 235  
 copyright, 2  
 core commands, 1334  
 count, 1154  
 count totalize, gating enable/disable, 309  
 count values, 246  
 count, averaged value math function, 384  
 count, averages, FFT function, 361  
 count, Edge Then Edge trigger, 1063  
 count, Nth edge of burst, 1067  
 counter, 302, 483  
 counter commands, 301  
 counter mode, 305  
 counter source, 307  
 counter totalize, clear, 308  
 counter totalize, gating polarity, 310  
 counter totalize, gating signal source, 311  
 counter totalize, slope, 312  
 counter, digits of resolution, 306  
 counter, enable/disable, 304  
 coupling, 1072  
 COUpling demo signal, 317  
 coupling for channels, 283  
 CRC format, SENT, 852  
 create automask, 573  
 crest factor, 550  
 CSV (Comma Separated Values) waveform data format, 717  
 cumulative edge activity, 1240  
 current harmonics analysis fail count, 626  
 current harmonics analysis results, save, 707  
 current harmonics analysis run count, 631  
 current harmonics analysis, apply, 623  
 current harmonics results data, 624  
 current harmonics results display, 625  
 current logic levels on digital channels, 207  
 current oscilloscope configuration, 187, 191, 195, 1030  
 current probe, 298, 357  
 CURRent segment waveform save option, 720  
 current source, 668  
 cursor display setting, X1, 444  
 cursor display setting, X2, 447  
 cursor display setting, Y1, 453  
 cursor display setting, Y2, 455  
 cursor mode, 443  
 cursor position, 442, 445, 448, 450, 454, 457  
 cursor readout, 1262, 1266, 1267  
 cursor reset conditions, 192, 1019  
 cursor source, 446, 449  
 cursor time, 1262, 1266, 1267  
 cursor units, X, 451, 452  
 cursor units, Y, 458, 459  
 cursor values, saving, 710  
 cursors track measurements, 516  
 cursors, how autoscale affects, 209  
 cursors, X1, X2, Y1, Y2, 441  
 cycle count base, FLEXray frame trigger, 790  
 cycle count repetition, FLEXray frame trigger, 791  
 cycle measured, 496, 499  
 cycle time, 506  
 cycles analyzed, number of, 652, 653

**D**

D- source, 1133

- D+ source, 1134  
 data, 816, 818, 815  
 data (waveform) maximum length, 719  
 data 2, 819  
 data acquisition types, 1148  
 data conversion, 1149  
 data format for transfer, 1150  
 data output order, 1153  
 data pattern length, 768, 836  
 data pattern, ARINC 429 trigger, 740, 953  
 data pattern, CAN trigger, 766  
 data point index, 1177  
 data points, 248  
 data record, measurement, 1160  
 data record, raw acquisition, 1160  
 data required to fill time buckets, 245  
 DATA source, I2S, 801  
 data source, setup and hold trigger, 1114  
 data structures, status reporting, 1301  
 data, saving and recalling, 330  
 date, calibration, 269  
 date, system, 1013  
 dB versus frequency, 405  
 DC coupling for edge trigger, 1072  
 DC input coupling for specified channel, 283  
 DC offset correction for integrate input, 401  
 DC RMS measured on waveform, 495, 536  
 DC waveform generator output, 1194  
 DCMotor demo signal, 316  
 DDE (Device Dependent Error) status bit, 183, 185  
 decision chart, status reporting, 1321  
 default conditions, 192, 1019  
 define channel labels, 287  
 define glitch trigger, 1083  
 define logic thresholds, 1242  
 define measurement, 486  
 define measurement source, 517  
 define trigger, 1084, 1102, 1103, 1105  
 defined as, 174  
 definite-length block query response, 64  
 definite-length block response data, 175  
 delay measured to calculate phase, 507  
 delay measurement, 486  
 delay measurements, 525  
 delay parameters for measurement, 488  
 delay time, Edge Then Edge trigger, 1062  
 DELay trigger commands, 1059  
 delay, how autoscale affects, 209  
 delayed time base, 1037  
 delayed window horizontal scale, 1045  
 delete mask, 583  
 delta time, 1262  
 delta voltage measurement, 1270  
 delta X cursor, 441  
 delta Y cursor, 441  
 demo, 313  
 DEMO commands, 313  
 demo signal, 315  
 demo signal function, 314  
 demo signal phase angle, 318  
 demo signals output control, 319  
 deskew for power measurements, 619  
 destination, remote command logging, 1024  
 detecting probe types, 1245, 1249  
 device-defined error queue clear, 181  
 DIFF source for function, 1253  
 differences from 4000 X-Series oscilloscope commands, 42  
 differential probe heads, 290  
 differential signal type, 294  
 differentiate math function, 405, 1167  
 digital channel commands, 322, 323, 324, 325, 327  
 digital channel data, 1151  
 digital channel labels, 342  
 digital channel order, 1247  
 digital channel source for glitch trigger, 1085  
 digital channels, 6  
 digital channels, activity and logic levels on, 207  
 digital channels, groups of, 601, 603, 605  
 digital pod, stop displaying, 214  
 digital reset conditions, 193, 1020  
 DIGItal <d> commands, 321  
 digitize channels, 215  
 DIGItize command, 56, 61, 1148  
 digits, 175  
 disable front panel, 1016  
 disable function, 1254  
 disabling calibration, 273  
 disabling channel display, 284  
 disabling status register bits, 182, 196  
 discontinued and obsolete commands, 1233  
 display annotation, 332  
 display annotation background, 333  
 display annotation color, 334  
 display annotation text, 335  
 display channel labels, 341  
 display clear, 338  
 DISPlay commands, 329  
 display commands introduction, 330  
 display connect, 1246  
 display date, 1013  
 display factors scaling, 289  
 display for channels, 284  
 display frequency span, 375, 396  
 display measurements, 476, 516  
 display mode, FFT, 365  
 display order, 1247  
 display persistence, 345  
 display reference, 1038, 1040  
 display reference waveforms, 1226  
 display reset conditions, 193, 1020  
 display serial number, 237  
 display vectors, 346  
 display wave position, 1247  
 display, FFT function, 364  
 display, lister, 437  
 display, measurement statistics on/off, 520  
 display, oscilloscope, 323, 345, 391, 603, 1014  
 display, serial decode bus, 726  
 displaying a baseline, 1058  
 displaying unsynchronized signal, 1058  
 divide math function, 405  
 DLC value in CAN FD trigger, 767  
 DNS IP, 50  
 domain, 50  
 domain, network printer, 427  
 driver, printer, 1259  
 DSO models, 6  
 duplicate mnemonics, 1339  
 duration, 1102, 1103, 1105  
 duration for glitch trigger, 1079, 1080, 1084  
 duration of power analysis, 654, 655, 656, 657, 658, 659  
 duration qualifier, trigger, 1102, 1103  
 duration triggering, 1048  
 duty cycle measurement, 57, 476, 496, 499  
 Duty Cycle, power modulation analysis, 640  
 DVM commands, 347  
 DVM displayed value, 349  
 DVM enable/disable, 350  
 DVM input source, 352  
 DVM mode, 351

## E

- EBURst trigger commands, 1066  
 ECL channel threshold, 1242  
 ECL threshold voltage for digital channels, 327  
 ECL trigger threshold voltage, 1288  
 edge activity, 1240  
 edge counter, Edge Then Edge trigger, 1063  
 edge counter, Nth edge of burst, 1067  
 edge coupling, 1072  
 edge fall time, 497  
 edge parameter for delay measurement, 488  
 edge preshoot measured, 509  
 edge rise time, 514  
 EDGE SEARch commands, 925  
 edge search slope, 926  
 edge search source, 927  
 edge slope, 1075  
 edge source, 1076  
 edge string for OR'ed edge trigger, 1097  
 EDGE trigger commands, 1071  
 edge triggering, 1048  
 edges (activity) on digital channels, 207  
 edges in measurement, 486  
 efficiency, 551  
 efficiency power analysis, apply, 620  
 elapsed time in mask test, 580

ellipsis, 175  
 enable channel labels, 341  
 enabling calibration, 273  
 enabling channel display, 284  
 enabling status register bits, 182, 196  
 end of string (EOS) terminator, 1336  
 end of text (EOT) terminator, 1336  
 end or identify (EOI), 1336  
 energy loss, 552  
 envelope math function, 406  
 EOI (end or identify), 1336  
 EOS (end of string) terminator, 1336  
 EOT (end of text) terminator, 1336  
 erase data, 338  
 erase measurements, 1261  
 erase screen, 1248  
 error counter (ARINC 429), 732  
 error counter (ARINC 429), reset, 733  
 error frame count (CAN), 748  
 error frame count (UART), 903  
 error messages, 1015, 1291  
 error number, 1015  
 error queue, 1015, 1312  
 error, measurement, 476  
 ESB (Event Status Bit), 197, 199  
 ESE (Standard Event Status Enable Register), 182, 1311  
 ESR (Standard Event Status Register), 184, 1310  
 ETE demo signal, 315  
 event status conditions occurred, 199  
 Event Status Enable Register (ESE), 182, 1311  
 Event Status Register (ESR), 184, 241, 1310  
 example code, :ACQuire:COMPLETE, 245  
 example code, :ACQuire:SEGmented, 251  
 example code, :ACQuire:TYPE, 256  
 example code, :AUToscale, 210  
 example code, :CHANnel<n>:LABEL, 287  
 example code, :CHANnel<n>:PROBe, 289  
 example code, :CHANnel<n>:RANGE, 296  
 example code, :DIGitize, 216  
 example code, :DISPlay:DATA, 339  
 example code, :DISPlay:LABEL, 341  
 example code, :DISPlay:ORDer, 1247  
 example code, :MEASure:PERiod, 518  
 example code, :MEASure:RESults, 511  
 example code, :MEASure:TEDGE, 526  
 example code, :MTEST, 569  
 example code, :POD<n>:THreshold, 605  
 example code, :RUN:/STOP, 236  
 example code, :SYSTem:SETUp, 1030  
 example code, :TIMEbase:DELay, 1287  
 example code, :TIMEbase:MODE, 1037  
 example code, :TIMEbase:RANGE, 1039  
 example code, :TIMEbase:REFERENCE, 1040  
 example code, :TRIGger:MODE, 1056  
 example code, :TRIGger:SLOPe, 1075  
 example code, :TRIGger:SOURce, 1076  
 example code, :VIEW and :BLANK, 242  
 example code, :WAVEform, 1168

example code, :WAVEform:DATA, 1155  
 example code, :WAVEform:POINTS, 1159  
 example code,  
   :WAVEform:PREamble, 1163  
 example code,  
   :WAVEform:SEGmented, 251  
 example code, \*RST, 194  
 example programs, 6, 1343  
 excursion delta for FFT peak search, 936  
 EXE (Execution Error) status bit, 183, 185  
 execution error detected in Standard Event Status, 185  
 exponential fall waveform generator output, 1195  
 exponential math function, 405  
 exponential notation, 174  
 exponential rise waveform generator output, 1195  
 extended video triggering license, 1127  
 external glitch trigger source, 1085  
 external range, 356  
 external trigger, 353, 355, 1076  
 EXTERNAL trigger commands, 353  
 EXTERNAL trigger level, 1073  
 external trigger probe attenuation factor, 355  
 external trigger probe sensing, 1249  
 EXTERNAL trigger source, 1076  
 external trigger units, 357

## F

fail count, current harmonics analysis, 626  
 fail/pass status (overall) for current harmonics analysis, 633  
 failed waveforms in mask test, 578  
 failure, self test, 201  
 fall time measurement, 476, 497  
 Fall Time, power modulation analysis, 640  
 falling edge count measurement, 500  
 falling pulse count measurement, 501  
 FAST data, SENT, 1171  
 Fast Fourier Transform (FFT)  
   functions, 362, 375, 377, 392, 396, 398, 405, 1253  
 FF values in waveform data, 1155  
 FFT (Fast Fourier Transform) functions, 362, 375, 377, 392, 396, 398, 405, 1253  
 FFT (Fast Fourier Transform)  
   operation, 1167  
 FFT display mode, 365  
 FFT function display, 364  
 FFT function, source input, 374  
 FFT vertical units, 376, 397  
 fifty ohm impedance, disable setting, 1022  
 filename for hardcopy, 1256  
 filename for recall, 687, 1192  
 filename for save, 698  
 filter for frequency reject, 1074  
 filter for high frequency reject, 1051  
 filter for noise reject, 1057

filter used to limit bandwidth, 282, 354  
 filters to Fast Fourier Transforms, 377, 398  
 filters, math, 405  
 fine horizontal adjustment (vernier), 1042  
 fine vertical adjustment (vernier), 299  
 finish pending device operations, 188  
 first point displayed, 1177  
 FLATtop window for amplitude measurements, 377, 398  
 FLEXray autoset for event trigger, 787  
 FLEXray autosearch, 777  
 FlexRay demo signal, 316  
 FlexRay frame counters, reset, 781  
 FLEXray SEARch commands, 966  
 FlexRay serial search, cycle, 967  
 FlexRay serial search, data, 968  
 FlexRay serial search, data length, 969  
 FlexRay serial search, frame, 970  
 FlexRay serial search, mode, 971  
 FLEXray source, 784  
 FLEXray trigger, 785  
 FLEXray trigger commands, 775  
 FM burst demo signal, 315  
 FM modulation type, waveform generator, 1210  
 force trigger, 1050  
 format, 1157, 1162  
 format (word), ARINC 429, 735  
 format for block data, 187  
 format for generic video, 1123, 1127  
 format for hardcopy, 1255  
 format for image, 701  
 format for waveform data, 717  
 FormattedIO488 object, 59  
 formfeed for hardcopy, 418, 422  
 formulas for data conversion, 1149  
 frame, 888  
 frame counters (CAN), error, 748  
 frame counters (CAN), overload, 749  
 frame counters (CAN), reset, 750  
 frame counters (CAN), total, 752  
 frame counters (FlexRay), null, 780, 782  
 frame counters (FlexRay), reset, 781  
 frame counters (FlexRay), total, 783  
 frame counters (UART), error, 903  
 frame counters (UART), reset, 904  
 frame counters (UART), Rx frames, 905  
 frame counters (UART), Tx frames, 906  
 frame ID, FLEXray BSS event trigger, 788  
 frame ID, FLEXray frame trigger, 792  
 frame type, FLEXray frame trigger, 793  
 framing, 886  
 frequency deviation, waveform generator FM modulation, 1202  
 frequency measurement, 57, 476, 498  
 frequency measurements with X cursors, 451  
 frequency resolution, 377, 398  
 frequency span of display, 375, 396  
 frequency versus dB, 405  
 Frequency, power modulation analysis, 640  
 front panel mode, 1058

front panel Single key, 238  
 front panel Stop key, 240  
 front-panel lock, 1016  
 FSK modulation type, waveform generator, 1210  
 FSK rate, waveform generator modulation, 1205  
 FT commands, 359  
 full-scale horizontal time, 1039, 1044  
 full-scale vertical axis defined, 371, 409  
 function, 362, 375, 377, 391, 392, 396, 398, 404, 405, 409, 410, 411, 1253, 1254  
 FUNCTION commands, 379  
 function memory, 239  
 function turned on or off, 1254  
 function, demo signal, 314  
 function, first source input, 413  
 function, second source input, 415  
 function, waveform generator, 1193  
 functions, 1167

**G**

g(t) source, first input channel, 1251  
 g(t) source, math operation, 1250  
 g(t) source, second input channel, 1252  
 gain for Ax + B math operation, 402  
 gated measurement window, 539  
 gateway IP, 50  
 gating, FFT math function, 369, 395  
 gaussian pulse waveform generator output, 1195  
 general SBUS<n> commands, 725  
 general SEARch commands, 920  
 general trigger commands, 1049  
 GENeric, 1123, 1127  
 generic video format, 1123, 1127  
 Generic video trigger, edge number, 1128  
 Generic video trigger, greater than sync pulse width time, 1131  
 Generic video trigger, horizontal sync control, 1129  
 Generic video trigger, horizontal sync pulse time, 1130  
 glitch demo signal, 314  
 glitch duration, 1084  
 glitch qualifier, 1083  
 GLITch SEARch commands, 928  
 glitch search, greater than value, 929  
 glitch search, less than value, 930  
 glitch search, polarity, 931  
 glitch search, qualifier, 932  
 glitch search, range, 933  
 glitch search, source, 934  
 glitch source, 1085  
 GLITch trigger commands, 1077  
 glitch trigger duration, 1079  
 glitch trigger polarity, 1082  
 glitch trigger source, 1079  
 GPIB interface, 50

graticule area for hardcopy print, 419  
 graticule colors, invert for hardcopy, 423, 1258  
 graticule colors, invert for image, 702  
 grayscale palette for hardcopy, 431  
 grayscale palette for image, 703  
 grayscaling on hardcopy, 1257  
 greater than qualifier, 1083  
 greater than time, 1079, 1084, 1102, 1105  
 greater than value for glitch search, 929  
 groups of digital channels, 601, 603, 605, 1242

**H**

HANNing window for frequency resolution, 377, 398  
 hardcopy, 235, 418  
 HARDcopy commands, 417  
 hardcopy factors, 421, 700  
 hardcopy filename, 1256  
 hardcopy format, 1255  
 hardcopy formfeed, 422  
 hardcopy grayscale, 1257  
 hardcopy invert graticule colors, 423, 1258  
 hardcopy layout, 424  
 hardcopy palette, 431  
 hardcopy print, area, 419  
 hardcopy printer driver, 1259  
 Hardware Event Condition Register (:HWERegister:CONDition), 219  
 Hardware Event Condition Register (:OPERegister:CONDition), 1318  
 Hardware Event Enable Register (HWEenable), 217  
 Hardware Event Event Register (:HWERegister[:EVENT]), 220, 1317  
 HARMonics demo signal, 317  
 head type, probe, 290  
 header, 1335  
 high pass filter math function, 406  
 high resolution acquisition type, 1149  
 high trigger level, 1054  
 high-frequency reject filter, 1051, 1074  
 high-level voltage, waveform generator, 1220  
 high-pass filter cutoff frequency, 399  
 high-resolution acquisition type, 244  
 hold time, setup and hold trigger, 1115  
 hold until operation complete, 188  
 holdoff time, 1052  
 holes in waveform data, 1155  
 hop frequency, waveform generator FSK modulation, 1204  
 horizontal adjustment, fine (vernier), 1042  
 horizontal position, 1043  
 horizontal scale, 1041, 1045  
 horizontal scaling, 1162  
 horizontal time, 1039, 1044, 1262

Host name softkey, 50  
 hostname, 50  
 HWEenable (Hardware Event Enable Register), 217  
 HWERegister:CONDition (Hardware Event Condition Register), 219, 1318  
 HWERegister[:EVENT] (Hardware Event Event Register), 220, 1317

**I**

I1080L50HZ, 1123, 1127  
 I1080L60HZ, 1123, 1127  
 I2C demo signal, 315  
 I2S alignment, 796  
 I2S audio channel, 805  
 I2S clock slope, 798  
 I2S CLOCk source, 800  
 I2S DATA source, 801  
 I2S demo signal, 316  
 I2S pattern data, 806  
 I2S pattern format, 808  
 I2S range, 809  
 I2S receiver width, 799  
 I2S SEARch commands, 972  
 I2S serial bus commands, 794  
 I2S serial decode base, 797  
 I2S serial search, audio channel, 973  
 I2S serial search, data, 975  
 I2S serial search, format, 976  
 I2S serial search, mode, 974  
 I2S serial search, range, 977  
 I2S transmit word size, 811  
 I2S trigger operator, 803  
 I2S triggering, 723  
 I2S word select (WS) low, 812  
 I2S word select (WS) source, 802  
 ID filter for CAN trigger, 765  
 id mode, 771  
 ID pattern, CAN trigger, 770  
 identification number, 186  
 identification of options, 189  
 identifier, LIN, 833  
 idle, 1068  
 idle state, SENT bus, 857  
 idle until operation complete, 188  
 IDN (Identification Number), 186  
 IEC 61000-3-2 standard for current harmonics analysis, 632  
 IEEE 488.2 standard, 180  
 IIC address, 817  
 IIC clock, 815  
 IIC data, 816, 818  
 IIC data 2, 819  
 IIC SEARch commands, 978  
 IIC serial decode address field size, 814  
 IIC serial search, address, 981  
 IIC serial search, data, 982  
 IIC serial search, data2, 983  
 IIC serial search, mode, 979  
 IIC serial search, qualifier, 984

IIC trigger commands, 813  
 IIC trigger qualifier, 820  
 IIC trigger type, 821  
 IIC triggering, 724  
 image format, 701  
 image invert graticule colors, 702  
 image memory, 239  
 image palette, 703  
 image, save, 699  
 image, save with inksaver, 702  
 impedance, 285  
 infinity representation, 1341  
 initial load current, transient response analysis, 680  
 initialization, 56, 59  
 initialize, 192, 1019  
 initialize label list, 342  
 initiate acquisition, 215  
 inksaver, save image with, 702  
 input coupling for channels, 283  
 input for integrate, DC offset correction, 401  
 input impedance for channels, 285, 1244  
 input inversion for specified channel, 286  
 input power, 554  
 inrush current, 558  
 inrush current analysis, 635, 636, 637  
 inrush current expected, 660  
 insert label, 287  
 installed options identified, 189  
 instruction header, 1335  
 instrument number, 186  
 instrument options identified, 189  
 instrument requests service, 199  
 instrument serial number, 237  
 instrument settings, 418  
 instrument status, 66  
 instrument type, 186  
 integrate DC offset correction, 401  
 integrate math function, 405, 1167  
 INTegrate source for function, 1253  
 intensity, waveform, 340  
 internal low-pass filter, 282, 354  
 introduction to :ACQuire commands, 243  
 introduction to :BUS<n> commands, 258  
 introduction to :CALibrate commands, 268  
 introduction to :CHANnel<n> commands, 281  
 introduction to :COUNter commands, 302  
 introduction to :DEMO commands, 313  
 introduction to :DIGItal<d> commands, 322  
 introduction to :DISPLAY commands, 330  
 introduction to :EXTer nal commands, 353  
 introduction to :FFT commands, 360  
 introduction to :FUNCTION commands, 383  
 introduction to :HARDcopy commands, 418  
 introduction to :LISTER commands, 435  
 introduction to :MARKer commands, 441  
 introduction to :MEASure commands, 476  
 introduction to :POD<n> commands, 601

introduction to :RECall commands, 684  
 introduction to :SAVE commands, 695  
 introduction to :SBUS commands, 723  
 introduction to :SYSTem commands, 1012  
 introduction to :TIMEbase commands, 1036  
 introduction to :TRIGger commands, 1047  
 introduction to :WAVEform commands, 1147  
 introduction to :WGEN<w> commands, 1184  
 introduction to :WMEMory<r> commands, 1223  
 introduction to common (\*) commands, 180  
 introduction to root(.) commands, 206  
 invert graticule colors for hardcopy, 423, 1258  
 invert graticule colors for image, 702  
 inverted masks, bind levels, 594  
 inverting input for channels, 286  
 IO library, referencing, 58  
 IP address, 50  
 IronPython, SCPI.NET example, 1462

## K

key disable, 1016  
 key press detected in Standard Event Status Register, 185  
 KEYSight demo signal, 317  
 Keysight Interactive IO application, 53  
 Keysight IO Control icon, 51  
 Keysight IO Libraries Suite, 6, 47, 58, 60  
 Keysight IO Libraries Suite, installing, 48  
 knob disable, 1016  
 known state, 192, 1019

## L

label, 1241  
 label command, bus, 264  
 label list, 287, 342  
 label reference waveforms, 1227  
 label, ARINC 429 trigger, 739, 743, 951  
 label, digital channel, 324  
 labels, 287, 341, 342  
 labels to store calibration information, 270  
 labels, specifying, 330  
 LAN instrument, 52  
 LAN interface, 49, 51  
 LAN Settings softkey, 50  
 landscape layout for hardcopy, 424  
 language for program examples, 55  
 layout for hardcopy, 424  
 leakage into peak spectrum, 377, 398  
 learn string, 187, 1030  
 least significant byte first, 1153  
 left reference, 1040  
 legal values for channel offset, 288

legal values for frequency span, 375, 396  
 legal values for offset, 404, 410  
 length for waveform data, 718  
 less than qualifier, 1083  
 less than time, 1080, 1084, 1103, 1105  
 less than value for glitch search, 930  
 level for trigger voltage, 1073, 1081  
 LF coupling, 1072  
 license information, 189  
 limits for line number, 1123  
 LIN acknowledge, 827  
 LIN baud rate, 828  
 LIN demo signal, 315  
 LIN identifier, 833  
 LIN pattern data, 834  
 LIN pattern format, 837  
 LIN SEARch commands, 985  
 LIN serial decode bus parity bits, 826  
 LIN serial search, data, 988  
 LIN serial search, data format, 990  
 LIN serial search, data length, 989  
 LIN serial search, frame ID, 986  
 LIN serial search, mode, 987  
 LIN source, 829  
 LIN standard, 830  
 LIN symbolic data display, 825  
 LIN symbolic data, recall, 688  
 LIN sync break, 831  
 LIN trigger, 832, 836  
 LIN trigger commands, 823  
 LIN trigger definition, 1284  
 LIN triggering, 724  
 line frequency setting for current harmonics analysis, 627  
 line glitch trigger source, 1085  
 line number for TV trigger, 1123  
 line terminator, 174  
 LINE trigger level, 1073  
 LINE trigger source, 1076  
 list of channel labels, 342  
 LISTer commands, 435  
 lister display, 437  
 lister time reference, 438  
 ln math function, 405  
 load utilization (CAN), 753  
 local lockout, 1016  
 lock, 1016  
 lock mask to signal, 585  
 lock, analog channel protection, 1022  
 lockout message, 1016  
 log file name, remote command logging, 1023, 1026  
 log math function, 405  
 logic level activity, 1240  
 long form, 1336  
 low frequency sine with glitch demo signal, 315  
 low pass filter math function, 406  
 low trigger level, 1055  
 lower threshold, 506  
 lower threshold voltage for measurement, 1260

lowercase characters in commands, 1335  
 low-frequency reject filter, 1074  
 low-level voltage, waveform generator, 1221  
 low-pass filter cutoff frequency, 400  
 low-pass filter used to limit bandwidth, 282, 354  
 LRN (Learn Device Setup), 187  
 lsbfirst, 1153

**M**

M1553 SEARch commands, 994  
 M1553 trigger commands, 841  
 M1553 trigger type, 847  
 magnify math function, 406  
 magnitude of occurrence, 527  
 main sweep range, 1043  
 main time base, 1287  
 main time base mode, 1037  
 making measurements, 476  
 MAN option for probe sense, 1245, 1249  
 manual cursor mode, 443  
 manufacturer string, 1017, 1018  
 MARKer commands, 439  
 marker mode, 454  
 marker position, 456  
 marker readout, 1266, 1267  
 marker set for voltage measurement, 1271, 1272  
 marker sets start time, 1263  
 marker time, 1262  
 markers for delta voltage measurement, 1270  
 markers track measurements, 516  
 markers, command overview, 441  
 markers, mode, 443  
 markers, time at start, 1267  
 markers, time at stop, 1266  
 markers, X delta, 442, 450  
 markers, X1 position, 445  
 markers, X1Y1 source, 446  
 markers, X2 position, 448  
 markers, X2Y2 source, 449  
 markers, Y delta, 457  
 markers, Y1 position, 454  
 markers, Y2 position, 456  
 mask, 182, 196  
 mask command, bus, 265  
 mask statistics, reset, 579  
 mask statistics, saving, 711  
 mask test commands, 567  
 Mask Test Event Enable Register (MTEenable), 221  
 mask test event event register, 223  
 Mask Test Event Register (.MTERegister[:EVENT]), 223, 1319  
 mask test run mode, 586  
 mask test termination conditions, 586  
 mask test, all channels, 572  
 mask test, enable/disable, 584

mask, delete, 583  
 mask, get as binary block data, 582  
 mask, load from binary block data, 582  
 mask, lock to signal, 585  
 mask, recall, 689  
 mask, save, 704, 705  
 masks, bind levels, 594  
 master summary status bit, 199  
 math filters, 405  
 math function, stop displaying, 214  
 math operators, 405  
 math transforms, 405  
 math visualizations, 406  
 MAV (Message Available), 181, 197, 199  
 max hold math function, 407  
 maximum duration, 1080, 1102, 1103  
 maximum math function, 406  
 maximum number of peaks for FFT peak search, 937  
 maximum position, 1038  
 maximum range for zoomed window, 1044  
 maximum scale for zoomed window, 1045  
 maximum vertical value measurement, 532  
 maximum vertical value, time of, 540, 1264  
 maximum waveform data length, 719  
 MEASure commands, 461  
 measure mask test failures, 587  
 measure overshoot, 503  
 measure period, 506  
 measure phase between channels, 507  
 MEASure power commands, 543  
 measure preshoot, 509  
 measure start voltage, 1271  
 measure stop voltage, 1272  
 measure value at a specified time, 537  
 measure value at top of waveform, 538  
 measurement error, 476  
 measurement record, 1160  
 measurement results, saving, 712  
 measurement setup, 476, 517  
 measurement source, 517  
 measurement statistics results, 511  
 measurement statistics, display on/off, 520  
 measurement trend math function, 407  
 measurement window, 539  
 measurements, AC RMS, 495, 536  
 measurements, area, 479, 490  
 measurements, average value, 492, 530  
 measurements, base value, 493, 531  
 measurements, built-in, 57  
 measurements, burst width, 481  
 measurements, clear, 482, 1261  
 measurements, command overview, 476  
 measurements, counter, 483  
 measurements, DC RMS, 495, 536  
 measurements, definition setup, 486  
 measurements, delay, 488  
 measurements, fall time, 497  
 measurements, falling edge count, 500  
 measurements, falling pulse count, 501  
 measurements, frequency, 498

measurements, how autoscale affects, 209  
 measurements, lower threshold level, 1260  
 measurements, maximum vertical value, 532  
 measurements, maximum vertical value, time of, 540, 1264  
 measurements, minimum vertical value, 533  
 measurements, minimum vertical value, time of, 541, 1265  
 measurements, negative duty cycle, 499  
 measurements, overshoot, 503  
 measurements, period, 506  
 measurements, phase, 507  
 measurements, positive duty cycle, 496  
 measurements, preshoot, 509  
 measurements, pulse width, negative, 502  
 measurements, pulse width, positive, 510  
 measurements, ratio of AC RMS values, 535  
 measurements, rise time, 514  
 measurements, rising edge count, 505  
 measurements, rising pulse count, 508  
 measurements, show, 516  
 measurements, snapshot all, 478  
 measurements, source channel, 517  
 measurements, standard deviation, 515  
 measurements, start marker time, 1266  
 measurements, stop marker time, 1267  
 measurements, thresholds, 1263  
 measurements, time between start and stop markers, 1262  
 measurements, time between trigger and edge, 525  
 measurements, time between trigger and vertical value, 527  
 measurements, time between trigger and voltage level, 1268  
 measurements, upper threshold value, 1269  
 measurements, vertical amplitude, 491, 529  
 measurements, vertical peak-to-peak, 494, 534  
 measurements, voltage difference, 1270  
 memory setup, 195, 1030  
 menu, display, 343  
 menu, system, 1286  
 message available bit, 199  
 message available bit clear, 181  
 message decode/trigerring format, SENT, 855  
 message displayed, 199  
 message error, 1291  
 message queue, 1309  
 message, CAN symbolic search, 963  
 message, CAN symbolic trigger, 772  
 message, LIN symbolic search, 991  
 message, LIN symbolic trigger, 838  
 messages ready, 199  
 midpoint of thresholds, 506  
 MIL-STD-1553 demo signal, 316

MIL-STD-1553 serial decode base, 843  
 MIL-STD-1553 serial search, data, 996  
 MIL-STD-1553 serial search, mode, 995  
 MIL-STD-1553 serial search, Remote Terminal Address, 997  
 MIL-STD-1553, dual demo signal, 316  
 min hold math function, 407  
 minimum duration, 1079, 1102, 1103, 1105  
 minimum math function, 407  
 minimum vertical value measurement, 533  
 minimum vertical value, time of, 541, 1265  
 MISO data pattern width, 892  
 MISO data pattern, SPI trigger, 891  
 MISO data source, SPI trigger, 889  
 MISO data, SPI, 1171  
 mixed-signal demo signals, 315  
 mixed-signal oscilloscopes, 6  
 mnemonics, duplicate, 1339  
 mode, 443, 1037, 1124  
 mode, serial decode, 727  
 model number, 186  
 models, oscilloscope, 3  
 modes for triggering, 1056  
 Modify softkey, 50  
 modulating signal frequency, waveform generator, 1201, 1203  
 modulation (waveform generator), enabling/disabling, 1209  
 modulation analysis, 638  
 modulation analysis source (voltage or current), 639  
 modulation analysis, type of, 640  
 modulation type, waveform generator, 1210  
 MOSI data pattern width, 894  
 MOSI data pattern, SPI trigger, 893  
 MOSI data source, SPI trigger, 890, 1285  
 most significant byte first, 1153  
 move cursors, 1266, 1267  
 msbfirst, 1153  
 MSG (Message), 197, 199  
 MSO models, 6  
 MSS (Master Summary Status), 199  
 MTEenable (Mask Test Event Enable Register), 221  
 MTERegister[:EVENT] (Mask Test Event Event Register), 223, 1319  
 MTEST commands, 567  
 multi-channel waveform data, save, 706  
 multiple commands, 1339  
 multiple queries, 65  
 multiplier value for SENT Fast Channel Signal, 863  
 multiply math function, 405, 1167  
 multiply math function as g(t) source, 1250

## N

N2820A high sensitivity current probe, 490, 491, 492, 493, 494, 495

N8900A Infinium Offline oscilloscope analysis software, 706  
 name channels, 287  
 name list, 342  
 natural logarithm math function, 405  
 negative glitch trigger polarity, 1082  
 negative pulse width, 502  
 negative pulse width measurement, 57  
 negative pulse width, power modulation analysis, 640  
 negative slope, 884, 1075  
 negative slope, Nth edge in burst, 1069  
 negative TV trigger polarity, 1125  
 network domain password, 428  
 network domain user name, 430  
 network printer address, 425  
 network printer domain, 427  
 network printer slot, 429  
 network printer, apply connection settings, 426  
 new line (NL) terminator, 174, 1336  
 new load current, transient response analysis, 681  
 NFC trigger commands, 1086  
 nibble order for SENT Fast Channel Signal, 865  
 NL (new line) terminator, 174, 1336  
 NMONotonic, dual demo signal, 316  
 noise floor, 674, 677  
 noise reject filter, 1057  
 noise waveform generator output, 1194  
 noise, adding to waveform generator output, 1208  
 noisy sine waveform demo signal, 314  
 non-core commands, 1334  
 non-interlaced GENeric mode, 1127  
 non-volatile memory, label list, 264, 324, 342  
 normal acquisition type, 243, 1148  
 normal trigger sweep mode, 1047  
 notices, 2  
 NR1 number format, 174  
 NR3 number format, 174  
 Nth edge burst trigger source, 1070  
 Nth edge burst triggering, 1048  
 Nth edge in a burst idle, 1068  
 Nth edge in burst slope, 1069  
 Nth edge of burst counter, 1067  
 Nth edge of Edge Then Edge trigger, 1063  
 NTSC, 1123, 1127  
 null frame count (FlexRay), 780  
 null offset, 674  
 NULL string, 1014  
 number format, 174  
 number of nibbles, SENT message, 858  
 number of points, 248, 1158, 1160  
 number of time buckets, 1158, 1160  
 numeric variables, 64  
 numeric variables, reading query results into multiple, 66  
 nwidth, 502

## O

obsolete and discontinued commands, 1233  
 obsolete commands, 1334  
 occurrence reported by magnitude, 1268  
 offset, 383  
 offset for Ax + B math operation, 403  
 offset value for channel voltage, 288  
 offset value for FFT function, 370, 372  
 offset value for selected function, 404, 410  
 offset value for SENT Fast Channel Signal, 864  
 offset, waveform generator, 1222  
 one values in waveform data, 1155  
 OPC (Operation Complete) command, 188  
 OPC (Operation Complete) status bit, 183, 185  
 OPEE (Operation Status Enable Register), 225  
 Open method, 59  
 operating configuration, 187, 1030  
 operating state, 195  
 operation complete, 188  
 operation status condition register, 227  
 Operation Status Condition Register (:OPERRegister:CONDition), 227, 1314  
 operation status conditions occurred, 199  
 Operation Status Enable Register (OPEE), 225  
 operation status event register, 229  
 Operation Status Event Register (:OPERRegister[:EVENT]), 229, 1313  
 operations for function, 405  
 operators, math, 405  
 OPERRegister:CONDition (Operation Status Condition Register), 227, 1314  
 OPERRegister[:EVENT] (Operation Status Event Register), 229, 1313  
 OPT (Option Identification), 189  
 optional syntax terms, 174  
 options, 189  
 OR trigger commands, 1096  
 order of digital channels on display, 1247  
 order of output, 1153  
 oscilloscope connection, opening, 59  
 oscilloscope connection, verifying, 51  
 oscilloscope external trigger, 353  
 oscilloscope models, 3  
 oscilloscope rate, 254  
 oscilloscope, connecting, 49  
 oscilloscope, initialization, 56  
 oscilloscope, operation, 6  
 oscilloscope, program structure, 56  
 oscilloscope, setting up, 49  
 oscilloscope, setup, 60  
 output control, demo signals, 319  
 output control, waveform generator, 1212  
 output load impedance, waveform generator, 1213  
 output messages ready, 199  
 output polarity, waveform generator, 1215

output power, 557  
 output queue, 188, 1308  
 output queue clear, 181  
 output ripple, 563  
 output ripple analysis, 650  
 output sequence, 1153  
 overall pass/fail status for current harmonics analysis, 633  
 overlapped commands, 1342  
 overload, 295  
 Overload Event Enable Register (OVL), 231  
 Overload Event Register (:OVLRegister), 1316  
 Overload Event Register (OVLR), 233  
 overload frame count (CAN), 749  
 overload protection, 231, 233  
 overshoot of waveform, 503  
 overshoot percent for transient response analysis, 661  
 overvoltage, 295  
 OVL (Overload Event Enable Register), 231  
 OVLR (Overload Event Register), 233  
 OVLR bit, 227, 229  
 OVLRegister (Overload Event Register), 1316

## P

P1080L24HZ, 1123, 1127  
 P1080L25HZ, 1123, 1127  
 P1080L50HZ, 1123, 1127  
 P1080L60HZ, 1123, 1127  
 P480L60HZ, 1123, 1127  
 P720L60HZ, 1123, 1127  
 PAL, 1123, 1127  
 palette for hardcopy, 431  
 palette for image, 703  
 PAL-M, 1123, 1127  
 parameters for delay measurement, 488  
 parametric measurements, 476  
 parity, 908  
 parity bits, LIN serial decode bus, 826  
 parser, 206, 1339  
 pass, self test, 201  
 pass/fail status (overall) for current harmonics analysis, 633  
 password, network domain, 428  
 path information, recall, 690  
 path information, save, 708  
 pattern, 817, 818, 819  
 pattern data, I2S, 806  
 pattern data, LIN, 834  
 pattern duration, 1079, 1080, 1102, 1103  
 pattern for pattern trigger, 1099  
 pattern format, I2S, 808  
 pattern format, LIN, 837  
 pattern length, 768, 836  
 PATTern trigger commands, 1098  
 pattern trigger format, 1101  
 pattern trigger qualifier, 1104

pattern triggering, 1048  
 pattern width, 892, 894  
 pause pulse, SENT messages, 859  
 peak current, 558  
 peak data, 1149  
 peak detect, 255  
 peak detect acquisition type, 244, 1149  
 PEAK SEARch commands, 935  
 peak-peak math function, 407  
 peak-to-peak vertical value measurement, 494, 534  
 pending operations, 188  
 percent of waveform overshoot, 503  
 percent thresholds, 486  
 period measured to calculate phase, 507  
 period measurement, 57, 476, 506  
 Period, power modulation analysis, 640  
 period, waveform generator, 1217  
 persistence, waveform, 330, 345  
 phase angle, demo signals, 318  
 phase measured between channels, 507  
 phase measurements, 525  
 phase measurements with X cursors, 451  
 phase shifted demo signals, 314  
 PLL Locked bit, 219  
 PNG format screen image data, 339  
 pod, 601, 603, 604, 605, 1167, 1242  
 POD commands, 601  
 POD data format, 1151  
 pod, stop displaying, 214  
 points, 248, 1158, 1160  
 points in waveform data, 1148  
 polarity, 909, 1125  
 polarity for glitch search, 931  
 polarity for glitch trigger, 1082  
 polarity, runt search, 941  
 polarity, runt trigger, 1107  
 polling synchronization with timeout, 1326  
 polling wait, 1324  
 PON (Power On) status bit, 183, 185  
 portrait layout for hardcopy, 424  
 position, 325, 448, 1038, 1043  
 position cursors, 1266, 1267  
 position in zoomed view, 1043  
 position waveforms, 1247  
 positive glitch trigger polarity, 1082  
 positive pulse width, 510  
 positive pulse width measurement, 57  
 positive pulse width, power modulation analysis, 640  
 positive slope, 884, 1075  
 positive slope, Nth edge in burst, 1069  
 positive TV trigger polarity, 1125  
 positive width, 510  
 power analysis, enabling, 622  
 POWer commands, 607  
 power factor, 553  
 power factor for IEC 61000-3-2 Standard Class C, 628  
 power loss, 559  
 power phase angle, 547  
 power quality analysis, 649  
 power supply rejection ratio (PSRR), 645, 646, 647, 648  
 preamble data, 1162  
 preamble metadata, 1147  
 predefined logic threshold, 1242  
 predefined threshold voltages, 1288  
 present working directory, recall operations, 690  
 present working directory, save operations, 708  
 preset conditions, 1019  
 preshoot measured on waveform, 509  
 previously stored configuration, 191  
 print command, 235  
 print job, start, 433  
 print mask test failures, 588  
 print query, 1281  
 printer driver for hardcopy, 1259  
 printer, active, 420  
 printing, 418  
 printing in grayscale, 1257  
 probe, 1073  
 probe attenuation affects channel voltage range, 296  
 probe attenuation factor (external trigger), 355  
 probe attenuation factor for selected channel, 289  
 probe head type, 290  
 probe ID, 291  
 probe sense for oscilloscope, 1245, 1249  
 probe skew value, 293, 1243  
 process sigma, mask test run, 591  
 program data, 1336  
 program data syntax rules, 1338  
 program initialization, 56  
 program message, 59, 180  
 program message syntax, 1335  
 program message terminator, 1336  
 program structure, 56  
 programming examples, 6, 1343  
 protecting against calibration, 273  
 protection, 231, 233, 295  
 protection lock, 1022  
 pulse waveform generator output, 1194  
 pulse width, 502, 510  
 pulse width duration trigger, 1079, 1080, 1084  
 pulse width measurement, 57, 476  
 pulse width trigger, 1057  
 pulse width trigger level, 1081  
 pulse width triggering, 1048  
 pulse width, waveform generator, 1197  
 pwidht, 510  
 Python, VISA COM example, 1370  
 Python, VISA example, 1417, 1423  
 PyVISA 1.5 and older, 1417  
 PyVISA 1.6 and newer, 1423

**Q**

qualifier, 1084  
 qualifier for glitch search, 932  
 qualifier, runt search, 942  
 qualifier, runt trigger, 1108  
 qualifier, transition search, 946  
 qualifier, transition trigger, 1118  
 qualifier, trigger duration, 1102, 1103  
 qualifier, trigger pattern, 1104  
 queries, multiple, 65  
 query error detected in Standard Event Status, 185  
 query responses, block data, 64  
 query responses, reading, 63  
 query results, reading into numeric variables, 64  
 query results, reading into string variables, 64  
 query return values, 1341  
 query setup, 418, 441, 476, 1030  
 query subsystem, 258, 322  
 querying setup, 281  
 querying the subsystem, 1048  
 queues, clearing, 1320  
 quick reference, commands, 69  
 quoted ASCII string, 175  
 QYE (Query Error) status bit, 183, 185

**R**

ramp symmetry, waveform generator, 1198  
 ramp symmetry, waveform generator modulating signal, 1207  
 ramp waveform generator output, 1194  
 range, 383, 1044  
 range for channels, 296  
 range for duration trigger, 1105  
 range for external trigger, 356  
 range for full-scale vertical axis, 371, 409  
 range for glitch search, 933  
 range for glitch trigger, 1084  
 range for time base, 1039  
 range of offset values, 288  
 range qualifier, 1083  
 range, I2S, 809  
 ranges, value, 175  
 rate, 254  
 ratio measurements with X cursors, 451  
 ratio measurements with Y cursors, 458  
 ratio of AC RMS values measured between channels, 535  
 Ratio, power modulation analysis, 640  
 raw acquisition record, 1160  
 RCL (Recall), 191  
 Rds (dynamic ON resistance) waveform, 673  
 Rds(on) power measurement, 560  
 Rds(on) value for conduction calculation, 675

reactive power, 561  
 read configuration, 187  
 ReadIEEEBlock method, 59, 63, 65  
 ReadList method, 59, 63  
 ReadNumber method, 59, 63  
 readout, 1262  
 ReadString method, 59, 63  
 real power, 562  
 Real Power source in Class D harmonics analysis, 629  
 real-time acquisition mode, 247  
 recall, 191, 684, 1030  
 recall arbitrary waveform, 685  
 recall CAN symbolic data, 686  
 RECall commands, 683  
 recall filename, 687, 1192  
 recall LIN symbolic data, 688  
 recall mask, 689  
 recall path information, 690  
 recall reference waveform, 692  
 recall setup, 691  
 recalling and saving data, 330  
 receiver width, I2S, 799  
 RECTangular window for transient signals, 377, 398  
 reference, 383, 1040  
 reference for time base, 1287  
 reference waveform save source, 721  
 reference waveform, recall, 692  
 reference waveform, save, 722  
 reference waveforms, clear, 1225  
 reference waveforms, display, 1226  
 reference waveforms, label, 1227  
 reference waveforms, save to, 1228  
 reference waveforms, skew, 1229  
 reference waveforms, Y offset, 1230  
 reference waveforms, Y range, 1231  
 reference waveforms, Y scale, 1232  
 reference, lister, 438  
 registers, 184, 191, 195, 208, 221, 223, 225, 227, 229, 231, 233  
 registers, clearing, 1320  
 reject filter, 1074  
 reject high frequency, 1051  
 reject noise, 1057  
 relative standard deviation, 524  
 remote command logging, enable/disable, 1023, 1027  
 remote control examples, 1343  
 Remote Terminal Address (RTA), M1553 trigger, 846  
 remove cursor information, 443  
 remove labels, 341  
 remove message from display, 1014  
 reorder channels, 209  
 repetitive acquisitions, 236  
 report errors, 1015  
 report transition, 525, 527  
 reporting status, 1299  
 reporting the setup, 1048  
 request service, 199  
 Request-for-OPC flag clear, 181  
 reset, 192  
 reset conditions, 192  
 reset defaults, waveform generator, 1218  
 reset mask statistics, 579  
 reset measurements, 338  
 resolution of printed copy, 1257  
 resource session object, 59  
 ResourceManager object, 59  
 restore configurations, 187, 191, 195, 1030  
 restore labels, 341  
 restore setup, 191  
 return values, query, 1341  
 returning acquisition type, 255  
 returning number of data points, 248  
 RF burst demo signal, 315  
 right reference, 1040  
 ringing pulse demo signal, 314  
 ripple (output) analysis, 650  
 ripple, output, 563  
 rise time measurement, 476  
 rise time of positive edge, 514  
 Rise Time, power modulation analysis, 640  
 rising edge count measurement, 505  
 rising pulse count measurement, 508  
 RMS - AC, power modulation analysis, 640  
 RMS value measurement, 495, 536  
 roll time base mode, 1037  
 root () commands, 203, 206  
 root level commands, 3  
 RQL (Request Control) status bit, 183, 185  
 RQS (Request Service), 199  
 RS-232/UART triggering, 724  
 RST (Reset), 192  
 rules, tree traversal, 1339  
 rules, truncation, 1336  
 run, 200, 236  
 Run bit, 227, 229  
 run count, current harmonics analysis, 631  
 run mode, mask test, 586  
 running configuration, 195, 1030  
 RUNT SEARch commands, 940  
 runt search polarity, 941  
 runt search qualifier, 942  
 runt search source, 943  
 runt search, pulse time, 944  
 RUNT trigger commands, 1106  
 runt trigger polarity, 1107  
 runt trigger qualifier, 1108  
 runt trigger source, 1109  
 runt trigger time, 1110  
 Rx frame count (UART), 905  
 Rx source, 910

**S**

sample rate, 3, 254  
 sampled data, 1246  
 sampled data points, 1155  
 SAV (Save), 195  
 save, 195, 695

save arbitrary waveform, 697  
 SAVE commands, 693  
 save current harmonics analysis results, 707  
 save filename, 698  
 save image, 699  
 save image with inksaver, 702  
 save mask, 704, 705  
 save mask test failures, 589  
 save path information, 708  
 save reference waveform, 722  
 save setup, 715  
 save to reference waveform location, 1228  
 save waveform data, 716  
 saved image, area, 1283  
 saving and recalling data, 330  
 SBUS A429 commands, 728  
 SBUS CAN commands, 745  
 SBUS commands, 723  
 SBUS I2S commands, 794  
 SBUS SENT commands, 848  
 SBUS<n> commands, general, 725  
 scale, 373, 411, 1041, 1045  
 scale factors output on hardcopy, 421, 700  
 scale for channels, 297  
 scale units for channels, 298  
 scale units for external trigger, 357  
 scaling display factors, 289  
 SCPI commands, 67  
 SCPI.NET example in C#, 1450  
 SCPI.NET example in IronPython, 1462  
 SCPI.NET example in Visual Basic .NET, 1456  
 SCPI.NET examples, 1450  
 scratch measurements, 1261  
 screen area for hardcopy print, 419  
 screen area for saved image, 1283  
 screen display of logged remote commands, enable/disable, 1025  
 screen image data, 339  
 SDI pattern, ARINC 429 trigger, 741, 954  
 SEARch commands, 919  
 SEARch commands, A429, 950  
 SEARch commands, CAN, 956  
 SEARch commands, EDGE, 925  
 SEARch commands, FLEXray, 966  
 SEARch commands, general, 920  
 SEARch commands, GLITch, 928  
 SEARch commands, I2S, 972  
 SEARch commands, IIC, 978  
 SEARch commands, LIN, 985  
 SEARch commands, M1553, 994  
 SEARch commands, PEAK, 935  
 SEARch commands, RUNT, 940  
 SEARch commands, SENT, 998  
 SEARch commands, SPI, 1003  
 SEARch commands, TRANSition, 945  
 SEARch commands, UART, 1007  
 search event (found) times, saving, 713  
 search for found event, 922  
 search mode, 923  
 search state, 924  
 search, edge slope, 926  
 search, edge source, 927  
 SECAM, 1123, 1127  
 seconds per division, 1041  
 segmented memory acquisition times, 714  
 segmented waveform save option, 720  
 segments, analyze, 249  
 segments, count of waveform, 1165  
 segments, setting number of memory, 250  
 segments, setting the index, 251  
 segments, time tag, 1166  
 select measurement channel, 517  
 self-test, 201  
 sensing a channel probe, 1245  
 sensing a external trigger probe, 1249  
 sensitivity of oscilloscope input, 289  
 SENT demo signal, 317  
 SENT enhanced serial message ID and data lengths, 879  
 SENT fast channel data search value, 999  
 SENT fast channel data trigger setting, 874  
 SENT FAST data, 1171  
 SENT input source, 869  
 SENT percent tolerance variation trigger, 880  
 SENT SEARch commands, 998  
 SENT search mode, 1000  
 SENT serial bus commands, 848  
 SENT signal length setting, 861  
 SENT signals display setting, 860  
 SENT slow channel data search value, 1001  
 SENT slow channel ID and data trigger data setting, 875  
 SENT slow channel ID search value, 1002  
 SENT slow channel ID trigger setting, 877  
 SENT SLOW data, 1171  
 SENT trigger mode, 872  
 SENT triggering, 724  
 sequential commands, 1342  
 serial clock, 815, 887  
 serial data, 816  
 serial decode bus, 723  
 serial decode bus display, 726  
 serial decode mode, 727  
 serial frame, 888  
 serial number, 237  
 service request, 199  
 Service Request Enable Register (SRE), 197, 1306  
 set center frequency, 362, 392  
 set cursors, 1266, 1267  
 set date, 1013  
 set time, 1032  
 set up oscilloscope, 49  
 setting digital display, 323  
 setting digital label, 264, 324  
 setting digital position, 325  
 setting digital threshold, 327  
 setting display, 391  
 setting external trigger level, 353  
 setting impedance for channels, 285  
 setting inversion for channels, 286  
 setting pod display, 603  
 setting pod size, 604  
 setting pod threshold, 605  
 settings, 191, 195  
 settings, instrument, 418  
 setup, 244, 258, 281, 322, 330, 418, 1030  
 setup and hold trigger clock source, 1113  
 setup and hold trigger data source, 1114  
 setup and hold trigger hold time, 1115  
 setup and hold trigger setup time, 1116  
 setup and hold trigger slope, 1112  
 setup configuration, 191, 195, 1030  
 setup defaults, 192, 1019  
 setup memory, 191  
 setup reported, 1048  
 setup time, setup and hold trigger, 1116  
 setup, recall, 691  
 setup, save, 715  
 shape of modulation signal, waveform generator, 1206  
 SHOOld trigger commands, 1111  
 short form, 5, 1336  
 show channel labels, 341  
 show measurements, 476, 516  
 SICL example in C, 1430  
 SICL example in Visual Basic, 1439  
 SICL examples, 1430  
 sidebar, display, 344  
 sigma, mask test run, 591  
 signal speed, ARINC 429, 738  
 signal type, 294  
 signal type, ARINC 429, 736  
 signal value, CAN symbolic search, 965  
 signal value, CAN symbolic trigger, 774  
 signal value, LIN symbolic search, 993  
 signal value, LIN symbolic trigger, 840  
 signal, CAN symbolic search, 964  
 signal, CAN symbolic trigger, 773  
 signal, LIN symbolic search, 992  
 signal, LIN symbolic trigger, 839  
 signed data, 1150  
 simple command headers, 1337  
 sine cardinal waveform generator output, 1195  
 sine waveform demo signal, 314  
 sine waveform generator output, 1193  
 single acquisition, 238  
 single-ended probe heads, 290  
 single-ended signal type, 294  
 single-shot demo signal, 314  
 single-shot DUT, synchronizing with, 1328  
 single-shot waveform generator output, 1216  
 size, 604  
 size, digital channels, 326  
 skew, 293, 1243  
 skew reference waveform, 1229  
 slew rate power analysis, 670  
 slope, 884, 1075  
 slope (direction) of waveform, 1268

slope not valid in TV trigger mode, 1075  
 slope parameter for delay measurement, 488  
 slope, arming edge, Edge Then Edge trigger, 1060  
 slope, Nth edge in burst, 1069  
 slope, setup and hold trigger, 1112  
 slope, transition search, 947  
 slope, transition trigger, 1119  
 slope, trigger edge, Edge Then Edge trigger, 1064  
 slot, network printer, 429  
 SLOW data, SENT, 1171  
 smoothing acquisition type, 1149  
 smoothing math function, 406  
 smoothing math function, number of points, 412  
 snapshot all measurement, 478  
 software version, 186  
 source, 517, 737, 761, 829  
 source (voltage or current) for slew rate analysis, 671  
 source channel, M1553, 844  
 source for function, 1253  
 source for glitch search, 934  
 source for Nth edge burst trigger, 1070  
 source for trigger, 1076  
 source for TV trigger, 1126  
 source function for FFT peak search, 938  
 source input for FFT function, 374  
 source input for function, first, 413  
 source input for function, second, 415  
 source, arming edge, Edge Then Edge trigger, 1061  
 source, automask, 574  
 source, FLEXray, 784  
 source, mask test, 599  
 source, NFC trigger, 1089  
 source, runt search, 943  
 source, runt trigger, 1109  
 source, save reference waveform, 721  
 source, transition trigger, 948, 1120  
 source, trigger edge, Edge Then Edge trigger, 1065  
 source, waveform, 1167  
 span, 405  
 span of frequency on display, 375, 396  
 Spec error counter (CAN), 751  
 specify measurement, 517  
 speed of ARINC 429 signal, 738  
 SPI, 884  
 SPI clock timeout, 885  
 SPI decode bit order, 883  
 SPI decode word width, 896  
 SPI demo signal, 316  
 SPI MISO data, 1171  
 SPI SEARch commands, 1003  
 SPI serial search, data, 1005  
 SPI serial search, data width, 1006  
 SPI serial search, mode, 1004  
 SPI trigger, 886, 892, 894  
 SPI trigger clock, 887

SPI trigger commands, 881  
 SPI trigger frame, 888  
 SPI trigger MISO data pattern, 891  
 SPI trigger MOSI data pattern, 893  
 SPI trigger type, 895  
 SPI trigger, MISO data source, 889  
 SPI trigger, MOSI data source, 890, 1285  
 SPI triggering, 724  
 square math function, 405  
 square root math function, 405  
 square wave duty cycle, waveform generator, 1199  
 square waveform generator output, 1193  
 SRE (Service Request Enable Register), 197, 1306  
 SRQ (Service Request interrupt), 217, 221, 225  
 SSM pattern, ARINC 429 trigger, 742, 955  
 standard deviation measured on waveform, 515  
 Standard Event Status Enable Register (ESE), 182, 1311  
 Standard Event Status Register (ESR), 184, 1310  
 standard for CAN FD decode, 756  
 standard for video, 1127  
 standard, LIN, 830  
 standard, NFC trigger, 1090  
 start acquisition, 200, 215, 236, 238  
 start and stop edges, 486  
 start cursor, 1266  
 start frequency, FFT math function, 367, 393  
 start measurement, 476  
 start print job, 433  
 start time, 1084, 1266  
 start time marker, 1263  
 starting bit for SENT Fast Channel Signal, 867  
 state memory, 195  
 state of instrument, 187, 1030  
 statistics increment, 521  
 statistics reset, 523  
 statistics results, 511  
 statistics, max count, 522  
 statistics, relative standard deviation, 524  
 statistics, type of, 519  
 status, 198, 239, 241  
 Status Byte Register (STB), 196, 198, 199, 1304  
 status data structure clear, 181  
 status registers, 66  
 status reporting, 1299  
 STB (Status Byte Register), 196, 198, 199, 1304  
 steady state output voltage expected, 665, 666, 667  
 step size for frequency span, 375, 396  
 stop, 215, 240  
 stop acquisition, 240  
 stop cursor, 1267  
 stop displaying channel, 214

stop displaying math function, 214  
 stop displaying pod, 214  
 stop frequency, FFT math function, 368, 394  
 stop on mask test failure, 590  
 stop time, 1084, 1267  
 storage, 195  
 store instrument setup, 187, 195  
 store setup, 195  
 storing calibration information, 270  
 string variables, 64  
 string variables, reading multiple query results into, 65  
 string variables, reading query results into multiple, 65  
 string, quoted ASCII, 175  
 subnet mask, 50  
 subsource, waveform source, 1171  
 subsystem commands, 3, 1339  
 subtract math function, 405, 1167  
 subtract math function as g(t) source, 1250  
 sweep mode, trigger, 1047, 1058  
 sweep speed set to fast to measure fall time, 497  
 sweep speed set to fast to measure rise time, 514  
 switch disable, 1016  
 switching level, current, 674  
 switching level, voltage, 677  
 switching loss per cycle, 549  
 switching loss power analysis, 672  
 sync break, LIN, 831  
 sync frame count (FlexRay), 782  
 syntax elements, 174  
 syntax rules, program data, 1338  
 syntax, optional terms, 174  
 syntax, program message, 1335  
 SYSTem commands, 1011  
 system commands, 1013, 1014, 1015, 1016, 1030, 1032  
 system commands introduction, 1012

## T

table of current harmonics results, 625  
 tdelta, 1262  
 tedge, 525  
 Tektronix probe model number, 292  
 telnet ports 5024 and 5025, 1155  
 Telnet sockets, 67  
 temporary message, 1014  
 TER (Trigger Event Register), 241, 1307  
 termination conditions, mask test, 586  
 test sigma, mask test run, 591  
 test, self, 201  
 text, writing to display, 1014  
 THD (total harmonics distortion), 634  
 threshold, 327, 605, 1242, 1288  
 threshold for FFT peak search, 939  
 threshold voltage (lower) for measurement, 1260

- threshold voltage (upper) for measurement, 1269  
 thresholds, 486, 1263  
 thresholds used to measure period, 506  
 thresholds, how autoscale affects, 209  
 time base, 1037, 1038, 1039, 1040, 1041, 1287  
 time base commands introduction, 1036  
 time base reset conditions, 193, 1020  
 time base window, 1043, 1044, 1045  
 time between arm and trigger, NFC arm and trigger event, 1088  
 time between points, 1262  
 time buckets, 245, 246  
 time delay, 1287  
 time delay, Edge Then Edge trigger, 1062  
 time delta, 1262  
 time difference between data points, 1175  
 time duration, 1084, 1102, 1103, 1105  
 time holdoff for trigger, 1052  
 time interval, 525, 527, 1262  
 time interval between trigger and occurrence, 1268  
 time marker sets start time, 1263  
 time measurements with X cursors, 451  
 time per division, 1039  
 time record, 377, 398  
 time reference, lister, 438  
 time specified, 537  
 time, calibration, 277  
 time, mask test run, 592  
 time, runt pulse search, 944  
 time, runt trigger, 1110  
 time, start marker, 1266  
 time, stop marker, 1267  
 time, system, 1032  
 time, transition search, 949  
 time, transition trigger, 1121  
 time/div, how autoscale affects, 209  
 time-at-max measurement, 1264  
 time-at-min measurement, 1265  
 TIMebase commands, 1035  
 timebase vernier, 1042  
 TIMebase:MODE, 62  
 time-ordered label list, 342  
 timeout enable, NFC arm and trigger event, 1094  
 timeout occurred, NFC arm and trigger event, 1093  
 timeout period, NFC arm and trigger event, 1095  
 timeout, SPI clock, 885  
 timing measurement, 476  
 title channels, 287  
 title, mask test, 600  
 tolerance for determining valid SENT sync pulses, 871  
 tolerance, automask, 576, 577  
 top of waveform value measured, 538  
 total frame count (CAN), 752  
 total frame count (FlexRay), 783  
 total harmonics distortion (THD), 634  
 total waveforms in mask test, 581  
 touchscreen on/off, 1033  
 trace memory, 239  
 track measurements, 516  
 transfer instrument state, 187, 1030  
 transforms, math, 405  
 transient response, 564  
 transient response analysis, 678, 679, 682  
 TRANSition SEARch commands, 945  
 transition search qualifier, 946  
 transition search slope, 947  
 transition search time, 949  
 transition trigger qualifier, 1118  
 transition trigger slope, 1119  
 transition trigger source, 948, 1120  
 transition trigger time, 1121  
 transmit word size, I2S, 811  
 transparent screen background, remote command logging, 1028  
 tree traversal rules, 1339  
 trend measurement, 416  
 TRG (Trigger), 197, 199, 200  
 TRIG OUT BNC, 271  
 trigger armed event register, 227, 229  
 trigger burst, UART, 913  
 trigger channel source, 1085, 1126  
 TRIGger commands, 1047  
 TRIGger commands, general, 1049  
 trigger data, UART, 914  
 TRIGger DELay commands, 1059  
 trigger duration, 1102, 1103  
 TRIGger EBURst commands, 1066  
 TRIGger EDGE commands, 1071  
 trigger edge coupling, 1072  
 trigger edge slope, 1075  
 trigger edge slope, Edge Then Edge trigger, 1064  
 trigger edge source, Edge Then Edge trigger, 1065  
 trigger event bit, 241  
 Trigger Event Register (TER), 1307  
 trigger event, NFC trigger, 1091  
 TRIGger FLEXray commands, 775  
 TRIGger GLITch commands, 1077  
 trigger holdoff, 1052  
 trigger idle, UART, 915  
 TRIGger IIC commands, 813  
 trigger level auto set up, 1053  
 trigger level constants, 289  
 trigger level voltage, 1073  
 trigger level, high, 1054  
 trigger level, low, 1055  
 TRIGger LIN commands, 823  
 TRIGger M1553 commands, 841  
 TRIGger NFC commands, 1086  
 trigger occurred, 199  
 TRIGger OR commands, 1096  
 TRIGger PATTern commands, 1098  
 trigger pattern qualifier, 1104  
 trigger qualifier, UART, 916  
 trigger reset conditions, 193, 1020  
 TRIGger RUNT commands, 1106  
 TRIGger SHOld commands, 1111  
 trigger SPI clock slope, 884  
 TRIGger SPI commands, 881  
 trigger status bit, 241  
 trigger sweep mode, 1047  
 TRIGger TV commands, 1117, 1122  
 trigger type, ARINC 429, 744, 952  
 trigger type, SPI, 895  
 trigger type, UART, 917  
 TRIGger UART commands, 897  
 TRIGger USB commands, 1132  
 TRIGger ZONE commands, 1137  
 trigger, ARINC 429 source, 737  
 trigger, CAN, 762  
 trigger, CAN FD sample point, 755  
 trigger, CAN pattern data length, 768  
 trigger, CAN pattern ID mode, 771  
 trigger, CAN sample point, 757  
 trigger, CAN signal baudrate, 758  
 trigger, CAN signal definition, 759  
 trigger, CAN source, 761  
 trigger, duration greater than, 1102  
 trigger, duration less than, 1103  
 trigger, duration range, 1105  
 trigger, edge coupling, 1072  
 trigger, edge level, 1073  
 trigger, edge reject, 1074  
 trigger, edge slope, 1075  
 trigger, edge source, 1076  
 trigger, FLEXray, 785  
 trigger, FLEXray error, 786  
 trigger, FLEXray event, 789  
 trigger, force a, 1050  
 trigger, glitch greater than, 1079  
 trigger, glitch less than, 1080  
 trigger, glitch level, 1081  
 trigger, glitch polarity, 1082  
 trigger, glitch qualifier, 1083  
 trigger, glitch range, 1084  
 trigger, glitch source, 1085  
 trigger, high frequency reject filter, 1051  
 trigger, holdoff, 1052  
 trigger, I2S, 803  
 trigger, I2S alignment, 796  
 trigger, I2S audio channel, 805  
 trigger, I2S clock slope, 798  
 trigger, I2S CLOCKsource, 800  
 trigger, I2S DATA source, 801  
 trigger, I2S pattern data, 806  
 trigger, I2S pattern format, 808  
 trigger, I2S range, 809  
 trigger, I2S receiver width, 799  
 trigger, I2S transmit word size, 811  
 trigger, I2S word select (WS) low, 812  
 trigger, I2S word select (WS) source, 802  
 trigger, IIC clock source, 815  
 trigger, IIC data source, 816  
 trigger, IIC pattern address, 817  
 trigger, IIC pattern data, 818  
 trigger, IIC pattern data 2, 819  
 trigger, IIC qualifier, 820  
 trigger, IIC signal baudrate, 828

trigger, IIC type, 821  
 trigger, LIN, 832  
 trigger, LIN pattern data, 834  
 trigger, LIN pattern data length, 836  
 trigger, LIN pattern format, 837  
 trigger, LIN sample point, 827  
 trigger, LIN signal definition, 1284  
 trigger, LIN source, 829  
 trigger, mode, 1056  
 trigger, noise reject filter, 1057  
 trigger, Nth edge burst source, 1070  
 trigger, Nth edge in burst slope, 1069  
 trigger, Nth edge of burst count, 1067  
 trigger, Nth edge of Edge Then Edge trigger, 1063  
 trigger, SPI clock slope, 884  
 trigger, SPI clock source, 887  
 trigger, SPI clock timeout, 885  
 trigger, SPI frame source, 888  
 trigger, SPI framing, 886  
 trigger, SPI pattern MISO width, 892  
 trigger, SPI pattern MOSI width, 894  
 trigger, sweep, 1058  
 trigger, threshold, 1288  
 trigger, TV line, 1123  
 trigger, TV mode, 1124, 1289  
 trigger, TV polarity, 1125  
 trigger, TV source, 1126  
 trigger, TV standard, 1127  
 trigger, UART base, 912  
 trigger, UART baudrate, 901  
 trigger, UART bit order, 902  
 trigger, UART parity, 908  
 trigger, UART polarity, 909  
 trigger, UART Rx source, 910  
 trigger, UART Tx source, 911  
 trigger, UART width, 918  
 trigger, USB, 1136  
 trigger, USB D- source, 1133  
 trigger, USB D+ source, 1134  
 trigger, USB speed, 1135  
 truncation rules, 1336  
 TST (Self Test), 201  
 tstart, 1266  
 tstop, 1267  
 TTL threshold voltage for digital channels, 327, 1242  
 TTL trigger threshold voltage, 1288  
 turn function on or off, 1254  
 turn off channel, 214  
 turn off channel labels, 341  
 turn off digital pod, 214  
 turn off math function, 214  
 turn off time, 555, 644  
 turn on channel labels, 341  
 turn on channel number display, 1247  
 turn on time, 556, 644  
 turn on/turn off time analysis, 641, 642, 643, 644  
 turning channel display on and off, 284  
 turning off/on function calculation, 391  
 turning vectors on or off, 1246

TV mode, 1124, 1289  
 TV trigger commands, 1117, 1122  
 TV trigger line number setting, 1123  
 TV trigger mode, 1126  
 TV trigger polarity, 1125  
 TV trigger standard setting, 1127  
 TV triggering, 1048  
 tvmode, 1289  
 Tx data, UART, 1171  
 Tx frame count (UART), 906  
 Tx source, 911  
 type of power being converted, efficiency measurement, 621

## U

UART base, 912  
 UART baud rate, 901  
 UART bit order, 902  
 UART frame counters, reset, 904  
 UART parity, 908  
 UART polarity, 909  
 UART Rx source, 910  
 UART SEARch commands, 1007  
 UART serial search, data, 1008  
 UART serial search, data qualifier, 1010  
 UART serial search, mode, 1009  
 UART trigger burst, 913  
 UART trigger commands, 897  
 UART trigger data, 914  
 UART trigger idle, 915  
 UART trigger qualifier, 916  
 UART trigger type, 917  
 UART Tx data, 1171  
 UART Tx source, 911  
 UART width, 918  
 UART/RS232 demo signal, 315  
 UART/RS-232 triggering, 724  
 units (vertical) for FFT, 376, 397  
 units per division, 297, 298, 357, 1041  
 units per division (vertical) for FFT function, 373  
 units per division (vertical) for function, 297, 411  
 units, automask, 575  
 units, X cursor, 451, 452  
 units, Y cursor, 458, 459  
 unsigned data, 1150  
 unsigned mode, 1173  
 upper threshold, 506  
 upper threshold voltage for measurement, 1269  
 uppercase characters in commands, 1335  
 URQ (User Request) status bit, 183, 185  
 USB (Device) interface, 49  
 USB source, 1133, 1134  
 USB speed, 1135  
 USB trigger, 1136  
 USB trigger commands, 1132  
 USB triggering, 1048  
 user defined channel labels, 287

user defined threshold, 1242  
 user event conditions occurred, 199  
 User Files directory, 690, 708  
 user name, network domain, 430  
 User's Guide, 6  
 user-defined Real Power in Class D harmonics analysis, 630  
 user-defined threshold voltage for digital channels, 327  
 user-defined trigger threshold, 1288  
 USR (User Event bit), 197, 199  
 utilization, CAN bus, 753

## V

valid command strings, 1335  
 valid pattern time, 1102, 1103  
 value, 527  
 value measured at base of waveform, 493, 531  
 value measured at specified time, 537  
 value measured at top of waveform, 538  
 value ranges, 175  
 values required to fill time buckets, 246  
 VBA, 58, 1344  
 Vce(sat) power measurement, 565  
 Vce(sat) value for conduction calculation, 676  
 vectors turned on or off, 1246  
 vectors, display, 346  
 vectors, turning on or off, 330  
 vernier, channel, 299  
 vernier, horizontal, 1042  
 vertical adjustment, fine (vernier), 299  
 vertical amplitude measurement, 491, 529  
 vertical axis defined by RANGe, 371, 409  
 vertical axis range for channels, 296  
 vertical offset for channels, 288  
 vertical peak-to-peak measured on waveform, 494, 534  
 vertical scale, 297, 373, 411  
 vertical scaling, 1162  
 vertical threshold, 1242  
 vertical units for FFT, 376, 397  
 vertical value at center screen, 370, 372, 404, 410  
 vertical value maximum measured on waveform, 532  
 vertical value measurements to calculate overshoot, 503  
 vertical value minimum measured on waveform, 533  
 video line to trigger on, 1123  
 video standard selection, 1127  
 view, 242, 1174, 1247  
 view turns function on or off, 1254  
 VISA COM example in C#, 1353  
 VISA COM example in Python, 1370  
 VISA COM example in Visual Basic, 1344  
 VISA COM example in Visual Basic .NET, 1362

VISA example in C, 1377  
 VISA example in C#, 1396  
 VISA example in Python, 1417, 1423  
 VISA example in Visual Basic, 1386  
 VISA example in Visual Basic .NET, 1407  
 VISA examples, 1344, 1377  
 Visual Basic .NET, SCPI.NET example, 1456  
 Visual Basic .NET, VISA COM example, 1362  
 Visual Basic .NET, VISA example, 1407  
 Visual Basic 6.0, 59  
 Visual Basic for Applications, 58, 1344  
 Visual Basic, SICL library example, 1439  
 Visual Basic, VISA COM example, 1344  
 Visual Basic, VISA example, 1386  
 visualizations, math, 406  
 voltage crossing reported or not found, 1268  
 voltage difference between data points, 1178  
 voltage difference measured, 1270  
 voltage level for active trigger, 1073  
 voltage marker used to measure waveform, 1271, 1272  
 voltage offset value for channels, 288  
 voltage probe, 298, 357  
 voltage ranges for channels, 296  
 voltage ranges for external trigger, 356  
 voltage source, 669  
 voltage threshold, 486  
 voltage, maximum expected input, 662, 663, 664

**W**

WAI (Wait To Continue), 202  
 wait, 202  
 wait for operation complete, 188  
 Wait Trig bit, 227, 229  
 warranty, 2  
 waveform base value measured, 493, 531  
 WAVEform command, 57  
 WAVEform commands, 1145  
 waveform data, 1147  
 waveform data format, 717  
 waveform data length, 718  
 waveform data length, maximum, 719  
 waveform data, save, 716  
 waveform generator, 1184  
 waveform generator amplitude, 1219  
 waveform generator function, 1193  
 waveform generator high-level voltage, 1220  
 waveform generator low-level voltage, 1221  
 waveform generator offset, 1222  
 waveform generator output control, 1212  
 waveform generator output load impedance, 1213  
 waveform generator output mode, 1214  
 waveform generator output polarity, 1215

waveform generator period, 1217  
 waveform generator pulse width, 1197  
 waveform generator ramp symmetry, 1198  
 waveform generator reset defaults, 1218  
 waveform generator square wave duty cycle, 1199  
 waveform introduction, 1147  
 waveform maximum vertical value measured, 532  
 waveform minimum vertical value measured, 533  
 waveform must cross voltage level to be an occurrence, 1268  
 WAVEform parameters, 62  
 waveform peak-to-peak vertical value measured, 494, 534  
 waveform period, 506  
 waveform persistence, 330  
 waveform RMS value measured, 495, 536  
 waveform save option for segments, 720  
 waveform source, 1167  
 waveform source subsource, 1171  
 waveform standard deviation value measured, 515  
 waveform vertical amplitude, 491, 529  
 waveform voltage measured at marker, 1271, 1272  
 waveform, byte order, 1153  
 waveform, count, 1154  
 waveform, data, 1155  
 waveform, format, 1157  
 waveform, points, 1158, 1160  
 waveform, preamble, 1162  
 waveform, type, 1172  
 waveform, unsigned, 1173  
 waveform, view, 1174  
 waveform, X increment, 1175  
 waveform, X origin, 1176  
 waveform, X reference, 1177  
 waveform, Y increment, 1178  
 waveform, Y origin, 1179  
 waveform, Y reference, 1180  
 WAVEform:FORMAT, 62  
 waveforms, mask test run, 593  
 Web control, 67  
 WGEN commands, 1181  
 WGEN trigger source, 1076  
 what's new, 35  
 width, 918, 1084  
 window, 1043, 1044, 1045  
 window time, 1039  
 window time base mode, 1037  
 window, measurement, 539  
 windows, 377, 398  
 windows as filters to Fast Fourier Transforms, 377, 398  
 windows for Fast Fourier Transform functions, 377, 398  
 WMEMory commands, 1223  
 word counter (ARINC 429), 734  
 word counter (ARINC 429), reset, 733  
 word format, 1157

word format for data transfer, 1150  
 word format, ARINC 429, 735  
 word select (WS) low, I2S trigger, 812  
 word select (WS) source, I2S, 802  
 word width, SPI decode, 896  
 write mode, remote command logging, 1023, 1029  
 write text to display, 1014  
 WriteIEEEBlock method, 59, 65  
 WriteList method, 59  
 WriteNumber method, 59  
 WriteString method, 59

**X**

X axis markers, 441  
 X cursor units, 451, 452  
 X delta, 442, 450  
 X delta, mask scaling, 596  
 X1 and X2 cursor value difference, 442, 450  
 X1 cursor, 441, 445, 446  
 X1, mask scaling, 595  
 X2 cursor, 441, 448, 449  
 X-axis functions, 1036  
 X-increment, 1175  
 X-of-max measurement, 540  
 X-of-min measurement, 541  
 X-origin, 1176  
 X-reference, 1177  
 X-Y mode, 1036, 1037

**Y**

Y axis markers, 441  
 Y cursor units, 458, 459  
 Y offset, reference waveform, 1230  
 Y range, reference waveform, 1231  
 Y scale, reference waveform, 1232  
 Y1 and Y2 cursor value difference, 457  
 Y1 cursor, 441, 446, 454, 457  
 Y1, mask scaling, 597  
 Y2 cursor, 441, 449, 456, 457  
 Y2, mask scaling, 598  
 Y-axis value, 1179  
 Y-increment, 1178  
 Y-origin, 1179, 1180  
 Y-reference, 1180

**Z**

zero values in waveform data, 1155  
 zone 1 or zone 2 mode, 1140  
 zone 1 or zone 2 placement, 1141  
 zone 1 or zone 2 state, 1143  
 zone 1 or zone 2 validity, 1142  
 zone qualified trigger source, 1138  
 zone qualified trigger state, 1139  
 ZONE trigger commands, 1137

zoomed time base, [1037](#)  
zoomed time base measurement  
    window, [539](#)  
zoomed time base mode, how autoscale  
    affects, [209](#)  
zoomed window horizontal scale, [1045](#)

