
InfiniiVision HD3-Series Mixed Signal Oscilloscopes

Notices

© Keysight Technologies, Inc. 2005-2025

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 10.21

Edition

December 2025

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. 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 HD3-Series mixed-signal oscilloscopes:

Table 1 HD3-Series Model Numbers, Bandwidths, Sample Rates

	HD302MSO	HD304MSO
Bandwidth	200 MHz standard, options for 350 MHz, 500 MHz, and 1 GHz	
Sample Rate	3.2 GSa/s	3.2 GSa/s
Memory Depth	20 Mpts standard, options for 50 Mpts and 100 Mpts	
Analog Channels	2	4
Logic Channels MSO	16	16

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

- **Chapter 1**, “What's New,” starting on page 41, describes programming command changes in the latest version of oscilloscope software.
- **Chapter 2**, “Setting Up,” starting on page 69, describes the steps you must take before you can program the oscilloscope.
- **Chapter 3**, “Getting Started,” starting on page 77, gives a general overview of oscilloscope program structure and shows how to program the oscilloscope using a few simple examples.
- **Chapter 4**, “Sequential (Blocking) vs. Overlapped Commands,” starting on page 91, describes these command types and how they affect the oscilloscope and synchronization.
- **Chapter 5**, “Commands Quick Reference,” starting on page 93, is a brief listing of the HD3-Series oscilloscope commands and syntax.

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

- **Chapter 6**, “Common (*) Commands,” starting on page 231, describes commands defined by the IEEE 488.2 standard that are common to all instruments.
- **Chapter 7**, “Root (:) Commands,” starting on page 259, describes commands that reside at the root level of the command tree and control many of the basic functions of the oscilloscope.
- **Chapter 8**, “:ACQuire Commands,” starting on page 283, describes commands for setting the parameters used when acquiring and storing data.
- **Chapter 9**, “:BUS<n> Commands,” starting on page 313, describes commands that control all oscilloscope functions associated with the digital channels bus display.

- **Chapter 10**, “:CALibrate Commands,” starting on page 323, describes utility commands for determining the state of the calibration factor protection button.
- **Chapter 11**, “:CGRade Commands,” starting on page 333, describes commands that control the color grade waveform analysis feature.
- **Chapter 12**, “:CHANnel<n> Commands,” starting on page 345, describes commands that control all oscilloscope functions associated with individual analog channels or groups of channels.
- **Chapter 13**, “:CLOCK Commands,” starting on page 379, describes clock recovery commands.
- **Chapter 14**, “:COUNter<c> Commands,” starting on page 385, describes commands that control the counter analysis feature.
- **Chapter 15**, “:DIGital<d> Commands,” starting on page 395, describes commands that control all oscilloscope functions associated with individual digital channels.
- **Chapter 16**, “:DISPlay Commands,” starting on page 405, describes commands that control how waveforms, graticule, and text are displayed and written on the screen.
- **Chapter 17**, “:DVM Commands,” starting on page 453, describes commands that control the digital voltmeter analysis feature.
- **Chapter 18**, “:EXTernal Trigger Commands,” starting on page 459, describes commands that control the input characteristics of the external trigger input.
- **Chapter 19**, “:FFT Commands,” starting on page 465, describes commands that control the FFT function in the oscilloscope.
- **Chapter 20**, “:FRANalysis Commands,” starting on page 497, describes commands that control oscilloscope functions associated with the Frequency Response Analysis (FRA) feature, which is available with the license-enabled built-in waveform generator).
- **Chapter 21**, “:FUNction<m> Commands,” starting on page 517, describes commands that control math waveforms.
- **Chapter 22**, “:HCOPy Commands,” starting on page 573, describes commands that set and query the selection of hardcopy device and formatting options.
- **Chapter 23**, “:HISTogram Commands,” starting on page 577, describes commands that control the histogram analysis feature.
- **Chapter 24**, “:JITTer Commands,” starting on page 593, describes commands that control the license-enabled jitter analysis feature.
- **Chapter 25**, “:LISTer Commands,” starting on page 613, describes commands that turn on/off the Lister display for decoded serial data and get the Lister data.
- **Chapter 26**, “:LTEST Commands,” starting on page 617, describes commands that control the Measurement Limit Test feature.
- **Chapter 27**, “:MARKer Commands,” starting on page 631, 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 28**, “:MEASure Commands,” starting on page 653, describes commands that select automatic measurements (and control markers).
- **Chapter 29**, “:MEASure Power Commands,” starting on page 769, describes measurement commands that are available when the power measurements and analysis application is licensed and enabled.
- **Chapter 30**, “:MTESt Commands,” starting on page 793, describes commands that control the mask test feature.
- **Chapter 31**, “:POD Commands,” starting on page 829, describes commands that control all oscilloscope functions associated with groups of digital channels.
- **Chapter 32**, “:POWer Commands,” starting on page 837, describes commands that control the license-enabled power measurement application.
- **Chapter 33**, “:RECall Commands,” starting on page 941, describes commands that recall previously saved oscilloscope setups, reference waveforms, or masks.
- **Chapter 34**, “:SAVE Commands,” starting on page 953, describes commands that save oscilloscope setups, screen images, and data.
- **Chapter 35**, “:SBUS<n> Commands,” starting on page 983, describes commands that control oscilloscope functions associated with the serial decode bus and serial triggering.
- **Chapter 36**, “:SEARch Commands,” starting on page 1173, describes commands that control oscilloscope functions associated with searching for waveform events.
- **Chapter 37**, “:STATus Commands,” starting on page 1283, describes commands that interact with status registers.
- **Chapter 38**, “:SYSTem Commands,” starting on page 1421, describes commands that control basic system functions of the oscilloscope.
- **Chapter 39**, “:TIMebase Commands,” starting on page 1467, describes commands that control all horizontal sweep functions.
- **Chapter 40**, “:TRIGger Commands,” starting on page 1479, describes commands that control the trigger modes and parameters for each trigger type.
- **Chapter 41**, “:WAVEform Commands,” starting on page 1553, describes commands that provide access to waveform data.
- **Chapter 42**, “:WGEN<w> Commands,” starting on page 1591, describes commands that control waveform generator (WAVEGEN license) functions and parameters.
- **Chapter 43**, “:WMEMory<r> Commands,” starting on page 1637, describes commands that control reference waveforms.
- **Chapter 44**, “Obsolete and Discontinued Commands,” starting on page 1647, describes obsolete commands which still work but have been replaced by newer commands and discontinued commands which are no longer supported.

- **Chapter 45**, “Error Messages,” starting on page 1661, 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 46**, “Status Reporting,” starting on page 1669, describes the oscilloscope's status registers and how to check the status of the instrument.
- **Chapter 47**, “Synchronizing Acquisitions,” starting on page 1685, describes how to wait for acquisitions to complete before querying measurement results or performing other operations with the captured data.
- **Chapter 48**, “More About Oscilloscope Commands,” starting on page 1707, contains additional information about oscilloscope programming commands.

Finally, there is a chapter that contains programming examples:

- **Chapter 49**, “Programming Examples,” starting on page 1717.

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 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 and search for "Connectivity Guide".
- For the latest versions of this and other manuals, see:
<http://www.keysight.com/find/HD3-Series-manual>

Contents

In This Book / 3

1 What's New

What's New in Version 10.21	/ 42
What's New in Version 10.20	/ 43
What's New in Version 10.15	/ 45
What's New in Version 10.10	/ 46
What's New in Version 10.07	/ 47
What's New in Version 10.06	/ 50
What's New in Version 10.05	/ 52
Version 10.00 at Introduction	/ 54
Command Differences From 3000G X-Series Oscilloscopes	/ 55

2 Setting Up

Step 1. Install Keysight IO Libraries Suite software	/ 70
Step 2. Connect and set up the oscilloscope	/ 71
Using the USB (Device) Interface	/ 71
Using the LAN Interface	/ 71
Step 3. Verify the oscilloscope connection	/ 73

3 Getting Started

Basic Oscilloscope Program Structure	/ 78
Initializing	/ 78
Capturing Data	/ 78
Analyzing Captured Data	/ 79

Programming the Oscilloscope /	80
Referencing the IO Library /	80
Opening the Oscilloscope Connection via the IO Library /	81
Initializing the Interface and the Oscilloscope /	81
Using :AUToscale to Automate Oscilloscope Setup /	82
Using Other Oscilloscope Setup Commands /	82
Capturing Data with the :DIGitize Command /	83
Reading Query Responses from the Oscilloscope /	85
Reading Query Results into String Variables /	85
Reading Query Results into Numeric Variables /	86
Reading Definite-Length Block Query Response Data /	86
Sending Multiple Queries and Reading Results /	87
Checking Instrument Status /	88
Other Ways of Sending Commands /	89
Telnet Sockets /	89
Sending SCPI Commands Using Browser Instrument IO /	89

4 Sequential (Blocking) vs. Overlapped Commands

5 Commands Quick Reference

Command Summary /	94
Syntax Elements /	227
Number Format /	227
<NL> (Line Terminator) /	227
[] (Optional Syntax Terms) /	227
{} (Braces) /	227
::= (Defined As) /	227
< > (Angle Brackets) /	228
... (Ellipsis) /	228
n,...,p (Value Ranges) /	228
d (Digits) /	228
Quoted ASCII String /	228
Definite-Length Block Response Data /	228

6 Common (*) Commands

*CLS (Clear Status) /	236
*ESE (Standard Event Status Enable) /	237
*ESR? (Standard Event Status Register) /	239
*IDN? (Identification Number) /	241
*LRN? (Learn Device Setup) /	242
*OPC (Operation Complete) /	243

*OPT? (Option Identification) / 244
*RCL (Recall) / 246
*RST (Reset) / 247
*SAV (Save) / 250
*SRE (Service Request Enable) / 251
*STB? (Read Status Byte) / 253
*TRG (Trigger) / 256
*TST? (Self Test) / 257
*WAI (Wait To Continue) / 258

7 Root (:) Commands

:ACTivity / 262
:AER? (Arm Event Register) / 263
:AUToscale / 265
:AUToscale:AMODE / 267
:AUToscale:CHANnels / 268
:AUToscale:FDEBug / 269
:BLANK / 270
:DIGitize / 271
:MODEl? / 273
:RSTate? / 274
:RUN / 275
:SERial? / 276
:SINGLE / 277
:STATus? / 278
:STOP / 279
:TER? (Trigger Event Register) / 280
:VIEW / 281

8 :ACquire Commands

:ACquire:AALias? / 286
:ACquire:COMPLete / 287
:ACquire:COUNt / 288
:ACquire:DAALias / 289
:ACquire:DIGitizer / 290
:ACquire:MANual / 292
:ACquire:MANual:POINTs? / 294
:ACquire:MANual:SRATE? / 295
:ACquire:MODE / 296
:ACquire:POINTs[:ANALog] / 297
:ACquire:POINTs[:ANALog]:AUTO / 298

:ACQuire:POINts:MAX? / 301
:ACQuire:SEGmented:ANALyze / 302
:ACQuire:SEGmented:COUNt / 303
:ACQuire:SEGmented:INDex / 304
:ACQuire:SRATe[:ANALog] / 307
:ACQuire:SRATe[:ANALog]:AUTO / 308
:ACQuire:TYPE / 311

9 :BUS<n> Commands

:BUS<n>:BIT<m> / 315
:BUS<n>:BITS / 316
:BUS<n>:CLEar / 318
:BUS<n>:DISPlay / 319
:BUS<n>:LABel / 320
:BUS<n>:MASK / 321

10 :CALibrate Commands

:CALibrate:DATE? / 325
:CALibrate:LABel / 326
:CALibrate:OUTPut / 327
:CALibrate:PROTected / 328
:CALibrate:STARt / 329
:CALibrate:STATus? / 330
:CALibrate:TEMPerature? / 331
:CALibrate:TIME? / 332

11 :CGRade Commands

:CGRade:ANALog:ALL / 335
:CGRade:INVert / 336
:CGRade:KEY:SOURce / 337
:CGRade:LEVelS / 338
:CGRade:LEVelS:DISPlay / 339
:CGRade:MATH:SOURce:STATe / 340
:CGRade:RESet / 341
:CGRade:SCHeme / 342
:CGRade:SOURce / 343
:CGRade:STATe / 344

12 :CHANnel<n> Commands

:CHANnel<n>:BANDwidth / 349
:CHANnel<n>:COUPling / 350

- :CHANnel<n>:DISPlay / 351
- :CHANnel<n>:IMPedance / 352
- :CHANnel<n>:INVert / 353
- :CHANnel<n>:LABel / 354
- :CHANnel<n>:OFFSet / 355
- :CHANnel<n>:PROBe / 356
- :CHANnel<n>:PROBe:BTN / 357
- :CHANnel<n>:PROBe:CALibration / 358
- :CHANnel<n>:PROBe:CALibration:VALidity / 359
- :CHANnel<n>:PROBe:EXTernal / 360
- :CHANnel<n>:PROBe:EXTernal:GAIN / 361
- :CHANnel<n>:PROBe:EXTernal:OFFSet / 362
- :CHANnel<n>:PROBe:EXTernal:UNITs / 364
- :CHANnel<n>:PROBe:HEAD[:TYPE] / 365
- :CHANnel<n>:PROBe:ID? / 366
- :CHANnel<n>:PROBe:MODE / 367
- :CHANnel<n>:PROBe:RSENse / 368
- :CHANnel<n>:PROBe:SKEW / 369
- :CHANnel<n>:PROBe:STATus? / 370
- :CHANnel<n>:PROBe:STYPe / 371
- :CHANnel<n>:PROBe:ZOOM / 372
- :CHANnel<n>:PROTection / 373
- :CHANnel<n>:RANGe / 374
- :CHANnel<n>:SCALE / 375
- :CHANnel<n>:UNITs / 376
- :CHANnel<n>:VERNier / 377

13 :CLOCK Commands

- :CLOCK / 381
- :CLOCK:METHod / 382

14 :COUNTER<c> Commands

- :COUNTER<c>:CURRENT? / 387
- :COUNTER<c>:ENABle / 388
- :COUNTER<c>:MODE / 389
- :COUNTER<c>:NDIGits / 390
- :COUNTER<c>:SOURce / 391
- :COUNTER<c>:TOTalize:CLEar / 392
- :COUNTER<c>:TOTalize:SLOPe / 393

15 :DIGital<d> Commands

:DIGital:ORDer:ASCending / 397
:DIGital:ORDer:DESCending / 398
:DIGital:ORDer[:SET] / 399
:DIGital<d>:DISPlay / 400
:DIGital<d>:LABel / 401
:DIGital<d>:SETup? / 402
:DIGital<d>:SIZE / 403
:DIGital<d>:THReshold / 404

16 :DISPlay Commands

:DISPlay:ANNotation<n> / 410
:DISPlay:ANNotation<n>:BACKground / 411
:DISPlay:ANNotation<n>:COLor / 412
:DISPlay:ANNotation<n>:GRID / 413
:DISPlay:ANNotation<n>:MODE / 414
:DISPlay:ANNotation<n>:SOURce / 415
:DISPlay:ANNotation<n>:TEXT / 416
:DISPlay:ANNotation<n>:XPOSition / 417
:DISPlay:ANNotation<n>:YPOSition / 418
:DISPlay:BACKlight / 419
:DISPlay:CLEar / 420
:DISPlay:CLOCK:IGRid / 421
:DISPlay:CLOCK:IGRid:XPOSition / 422
:DISPlay:CLOCK:IGRid:YPOSition / 423
:DISPlay:CLOCK[:STATe] / 424
:DISPlay:DATA? / 425
:DISPlay:DIALog:CLOSe / 426
:DISPlay:DIALog:TRANSPARENT / 427
:DISPlay:GRATicule:ALABels / 428
:DISPlay:GRATicule:ALABels:DUAL / 429
:DISPlay:GRATicule:ALABels:IGRid / 430
:DISPlay:GRATicule:COUNt / 431
:DISPlay:GRATicule:FSCreen / 432
:DISPlay:GRATicule:INTensity / 433
:DISPlay:GRATicule:LAYout / 434
:DISPlay:GRATicule:SET / 435
:DISPlay:GRATicule:SOURce? / 436
:DISPlay:INTensity:WAVEform / 437
:DISPlay:LABel / 438
:DISPlay:LABList / 439

:DISPlay:MESSage:CLEar / 440
:DISPlay:PERSistence / 441
:DISPlay:PERSistence:CLEar / 442
:DISPlay:RESults:CATalog? / 443
:DISPlay:RESults:LAYout / 444
:DISPlay:RESults:LAYout:LIST[:SElect] / 445
:DISPlay:RESults:LAYout:TAB:LEFT[:SElect] / 446
:DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] / 447
:DISPlay:RESults:SIZE / 448
:DISPlay:TRANSPARENT / 449
:DISPlay:VECTors / 450
:DISPlay:XYColor / 451

17 :DVM Commands

:DVM:ARANge / 454
:DVM:CURREnt? / 455
:DVM:ENABle / 456
:DVM:MODE / 457
:DVM:SOURce / 458

18 :EXTeRnal Trigger Commands

:EXTeRnal:BWLimit / 460
:EXTeRnal:PROBe / 461
:EXTeRnal:RANGe / 462
:EXTeRnal:UNITs / 463

19 :FFT Commands

:FFT:AVERage:COUNt / 468
:FFT:BSIZe? / 469
:FFT:CENTer / 470
:FFT:CLEar / 471
:FFT:DETection:POINts / 472
:FFT:DETection:TYPE / 473
:FFT:DISPlay / 474
:FFT:DMODE / 475
:FFT:FREQuency:STARt / 476
:FFT:FREQuency:STOP / 477
:FFT:GATE / 478
:FFT:OFFSet / 480
:FFT:RANGe / 481
:FFT:RBWidth? / 482

:FFT:READout / 483
:FFT:REFeRence / 484
:FFT:SCALe / 485
:FFT:SOURce / 486
:FFT:SPAN / 487
:FFT:SRATe? / 488
:FFT:VTYPe / 489
:FFT:WINDow / 490
:FFTMax:OFFSet / 492
:FFTMax:RESet / 493
:FFTMax:SCALe / 494
:FFTMax:STATe / 495

20 :FRANalysis Commands

:FRANalysis:DATA? / 500
:FRANalysis:ENABle / 501
:FRANalysis:FREQuency:MODE / 502
:FRANalysis:FREQuency:SINGle / 503
:FRANalysis:FREQuency:STARt / 504
:FRANalysis:FREQuency:STOP / 505
:FRANalysis:PPDecade / 506
:FRANalysis:RUN / 507
:FRANalysis:SOURce:INPut / 508
:FRANalysis:SOURce:OUTPut / 509
:FRANalysis:SWEep:POINts / 510
:FRANalysis:TRACe / 511
:FRANalysis:WGEN:LOAD / 512
:FRANalysis:WGEN:VOLTage / 513
:FRANalysis:WGEN:VOLTage:PROFile / 515

21 :FUNCTion<m> Commands

:FUNCTion<m>:AUTosetup / 524
:FUNCTion<m>:AUTosetup:ENABle / 525
:FUNCTion<m>:AVERage:COUNt / 526
:FUNCTion<m>:BUS:CLOCK / 527
:FUNCTion<m>:BUS:SLOPe / 528
:FUNCTion<m>:BUS:YINCrement / 529
:FUNCTion<m>:BUS:YORigin / 530
:FUNCTion<m>:BUS:YUNits / 531
:FUNCTion<m>:CLEar / 532
:FUNCTion<m>:DISPlay / 533

:FUNCTION<m>[:FFT]:BSIZE? / 534
 :FUNCTION<m>[:FFT]:CENTER / 535
 :FUNCTION<m>[:FFT]:DETECTION:POINTS / 536
 :FUNCTION<m>[:FFT]:DETECTION:TYPE / 537
 :FUNCTION<m>[:FFT]:FREQUENCY:START / 538
 :FUNCTION<m>[:FFT]:FREQUENCY:STOP / 539
 :FUNCTION<m>:FFT:GATE / 540
 :FUNCTION<m>[:FFT]:PHASE:REFERENCE / 542
 :FUNCTION<m>[:FFT]:RBWIDTH? / 543
 :FUNCTION<m>[:FFT]:READOUT<n> / 544
 :FUNCTION<m>[:FFT]:SPAN / 545
 :FUNCTION<m>[:FFT]:SRATE? / 546
 :FUNCTION<m>[:FFT]:VTYPE / 547
 :FUNCTION<m>[:FFT]:WINDOW / 548
 :FUNCTION<m>:FREQUENCY:BANDPASS:CENTER / 549
 :FUNCTION<m>:FREQUENCY:BANDPASS:WIDTH / 550
 :FUNCTION<m>:FREQUENCY:HIGHPASS / 551
 :FUNCTION<m>:FREQUENCY:LOWPASS / 552
 :FUNCTION<m>:GATE / 553
 :FUNCTION<m>:INTEGRATE:IICONDITION / 555
 :FUNCTION<m>:INTEGRATE:IOFFSET / 556
 :FUNCTION<m>:LABEL / 557
 :FUNCTION<m>:LINEAR:GAIN / 558
 :FUNCTION<m>:LINEAR:OFFSET / 559
 :FUNCTION<m>:OFFSET / 560
 :FUNCTION<m>:OPERATION / 561
 :FUNCTION<m>:RANGE / 565
 :FUNCTION<m>:REFERENCE / 566
 :FUNCTION<m>:SCALE / 567
 :FUNCTION<m>:SMOOTH:POINTS / 568
 :FUNCTION<m>:SOURCE1 / 569
 :FUNCTION<m>:SOURCE2 / 571
 :FUNCTION<m>:TEND:NMEASUREMENT / 572

22 :HCOPY Commands

:HCOPY:SDUMP:DATA? / 574
 :HCOPY:SDUMP[:DATA]:FORMAT / 575

23 :HISTOGRAM Commands

:HISTOGRAM:AXIS / 580
 :HISTOGRAM:DISPLAY / 581

:HISTogram:MEASurement / 582
:HISTogram:MODE / 583
:HISTogram:RESet / 584
:HISTogram:SIZE / 585
:HISTogram:TYPE / 586
:HISTogram:WINDow:BLIMit / 587
:HISTogram:WINDow:LLIMit / 588
:HISTogram:WINDow:RLIMit / 589
:HISTogram:WINDow:SOURce / 590
:HISTogram:WINDow:TLIMit / 591

24 :JITter Commands

:JITter:HISTogram:APPLy / 595
:JITter:MEASurement / 596
:JITter:SPECTrum:APPLy / 598
:JITter:SPECTrum:HORizontal:POSition / 599
:JITter:SPECTrum:HORizontal:RANGe / 600
:JITter:SPECTrum:VERTical:OFFSet / 601
:JITter:SPECTrum:VERTical:RANGe / 602
:JITter:SPECTrum:VERTical:TYPE / 603
:JITter:SPECTrum:WINDow / 604
:JITter:STATe / 606
:JITter:TREND:APPLy / 608
:JITter:TREND:SMOoth:APPLy / 609
:JITter:TREND:SMOoth:POINts / 610
:JITter:TREND:VERTical:OFFSet / 611
:JITter:TREND:VERTical:RANGe / 612

25 :LISTer Commands

:LISTer:DATA? / 614
:LISTer:DISPlay / 615
:LISTer:REFerence / 616

26 :LTEST Commands

:LTEST:COPY / 619
:LTEST:COPY:ALL / 620
:LTEST:COPY:MARGin / 621
:LTEST:ENABle / 622
:LTEST:FAIL / 623
:LTEST:LLIMit / 624
:LTEST:MEASurement / 625

:LTEST:RESults? / 626
:LTEST:RUMode:SOfailure / 627
:LTEST:TEST / 628
:LTEST:ULIMit / 629

27 :MARKer Commands

:MARKer:DYDX? / 634
:MARKer:MODE / 635
:MARKer:X1:DISPlay / 636
:MARKer:X1Position / 637
:MARKer:X1Y1source / 638
:MARKer:X2:DISPlay / 639
:MARKer:X2Position / 640
:MARKer:X2Y2source / 641
:MARKer:XDELta? / 642
:MARKer:XUNits / 643
:MARKer:XUNits:USE / 644
:MARKer:Y1:DISPlay / 645
:MARKer:Y1Position / 646
:MARKer:Y2:DISPlay / 647
:MARKer:Y2Position / 648
:MARKer:YDELta? / 649
:MARKer:YUNits / 650
:MARKer:YUNits:USE / 651

28 :MEASure Commands

:MEASure:AREA / 673
:MEASure:BRATe / 674
:MEASure:BWIDth / 675
:MEASure:CLEar / 676
:MEASure:COUNter / 677
:MEASure:CTCNwidth / 679
:MEASure:CTCPwidth / 681
:MEASure:DELay / 683
:MEASure:DELay:DEFine / 685
:MEASure:DUTYcycle / 687
:MEASure:FALLtime / 688
:MEASure:FFT:ACPR / 689
:MEASure:FFT:CPOWer / 690
:MEASure:FFT:OBW / 691
:MEASure:FFT:THD / 692

:MEASure:FREQuency / 693
:MEASure:HISTogram:BWIDth? / 694
:MEASure:HISTogram:HITS? / 695
:MEASure:HISTogram:M1S? / 696
:MEASure:HISTogram:M2S? / 697
:MEASure:HISTogram:M3S? / 698
:MEASure:HISTogram:MAXimum? / 699
:MEASure:HISTogram:MEAN? / 700
:MEASure:HISTogram:MEDian? / 701
:MEASure:HISTogram:MINimum? / 702
:MEASure:HISTogram:MODE? / 703
:MEASure:HISTogram:PEAK? / 704
:MEASure:HISTogram:PPEak? / 705
:MEASure:HISTogram:SDEViation? / 706
:MEASure:HYSTeresis / 707
:MEASure:NCJitter / 708
:MEASure:NDUTy / 710
:MEASure:NEDGes / 711
:MEASure:NPERiod / 712
:MEASure:NPULses / 713
:MEASure:NWIDth / 714
:MEASure:OVERshoot / 715
:MEASure:PEDGes / 717
:MEASure:PERiod / 718
:MEASure:PHASe / 719
:MEASure:PPULses / 720
:MEASure:PREShoot / 721
:MEASure:PWIDth / 722
:MEASure:QUICK / 723
:MEASure:RESults? / 724
:MEASure:RISetime / 728
:MEASure:SDEViation / 729
:MEASure:SHOW / 730
:MEASure:SLEWrate / 731
:MEASure:SOURce / 733
:MEASure:STATistics / 736
:MEASure:STATistics:DISPlay / 737
:MEASure:STATistics:INCRement / 738
:MEASure:STATistics:MCOunt / 739
:MEASure:STATistics:RESet / 740
:MEASure:STATistics:RSDeviation / 741
:MEASure:TEDGE / 742

:MEASure:THResholds:ABSolute / 746
 :MEASure:THResholds:METHod / 747
 :MEASure:THResholds:PERCent / 748
 :MEASure:THResholds:STANdard / 749
 :MEASure:TIEClock / 750
 :MEASure:TIEData / 752
 :MEASure:TVALue? / 754
 :MEASure:VAMPlitude / 756
 :MEASure:VAverage / 757
 :MEASure:VBASe / 758
 :MEASure:VMAX / 759
 :MEASure:VMIN / 760
 :MEASure:VPP / 761
 :MEASure:VRATio / 762
 :MEASure:VRMS / 763
 :MEASure:VTOP / 764
 :MEASure:WINDow / 765
 :MEASure:XMAX / 766
 :MEASure:XMIN / 767
 :MEASure:YATX / 768

29 :MEASure Power Commands

:MEASure:ANGLE / 774
 :MEASure:APParent / 775
 :MEASure:CPLoss / 776
 :MEASure:CRESt / 777
 :MEASure:EFFiciency / 778
 :MEASure:ELOSs / 779
 :MEASure:FACTor / 780
 :MEASure:IPOWER / 781
 :MEASure:OFFTime / 782
 :MEASure:ONTime / 783
 :MEASure:OPOWER / 784
 :MEASure:PCURrent / 785
 :MEASure:PLOSs / 786
 :MEASure:RDSon / 787
 :MEASure:REACTive / 788
 :MEASure:REAL / 789
 :MEASure:RIPple / 790
 :MEASure:TRESponse / 791
 :MEASure:VCESat / 792

30 :MTESt Commands

:MTESt:ALL / 799
:MTESt:AMASk:CREate / 800
:MTESt:AMASk:SOURce / 801
:MTESt:AMASk:UNITs / 802
:MTESt:AMASk:XDELta / 803
:MTESt:AMASk:YDELta / 804
:MTESt:COUNt:FWAVeforms? / 805
:MTESt:COUNt:RESet / 806
:MTESt:COUNt:TIME? / 807
:MTESt:COUNt:WAVeforms? / 808
:MTESt:DATA / 809
:MTESt:DELeTe / 810
:MTESt:ENABle / 811
:MTESt:LOCK / 812
:MTESt:RMODe / 813
:MTESt:RMODe:FACTION:MEASure / 814
:MTESt:RMODe:FACTION:PRINt / 815
:MTESt:RMODe:FACTION:SAVE / 816
:MTESt:RMODe:FACTION:STOP / 817
:MTESt:RMODe:SIGMa / 818
:MTESt:RMODe:TIME / 819
:MTESt:RMODe:WAVeforms / 820
:MTESt:SCALE:BIND / 821
:MTESt:SCALE:X1 / 822
:MTESt:SCALE:XDELta / 823
:MTESt:SCALE:Y1 / 824
:MTESt:SCALE:Y2 / 825
:MTESt:SOURce / 826
:MTESt:TITLe? / 827

31 :POD Commands

:POD<n>:DISPlay / 831
:POD<n>:NIBBle<n>:THReshold / 832
:POD<n>:SIZE / 834
:POD<n>:THReshold / 835

32 :POWer Commands

:POWer:CLResponse? / 845
:POWer:CLResponse:APPLy / 846
:POWer:CLResponse:DATA? / 847

:POWer:CLResponse:DATA:GMARgin? / 848
:POWer:CLResponse:DATA:GMARgin:FREQuency? / 849
:POWer:CLResponse:DATA:PMARgin? / 850
:POWer:CLResponse:DATA:PMARgin:FREQuency? / 851
:POWer:CLResponse:FREQuency:MODE / 852
:POWer:CLResponse:FREQuency:SINGle / 853
:POWer:CLResponse:FREQuency:STARt / 854
:POWer:CLResponse:FREQuency:STOP / 855
:POWer:CLResponse:PPDecade / 856
:POWer:CLResponse:SOURce:INPut / 857
:POWer:CLResponse:SOURce:OUTPut / 858
:POWer:CLResponse:TRACe / 859
:POWer:CLResponse:WGEN:LOAD / 860
:POWer:CLResponse:WGEN:VOLTage / 861
:POWer:CLResponse:WGEN:VOLTage:PROFile / 862
:POWer:DESKew / 863
:POWer:EFFiciency:APPLy / 864
:POWer:EFFiciency:TYPE / 865
:POWer:ENABle / 866
:POWer:HARMonics:APPLy / 867
:POWer:HARMonics:DATA? / 868
:POWer:HARMonics:DISPlay / 869
:POWer:HARMonics:FAILcount? / 870
:POWer:HARMonics:LINE / 871
:POWer:HARMonics:POWerfactor? / 872
:POWer:HARMonics:RPOWer / 873
:POWer:HARMonics:RPOWer:USER / 874
:POWer:HARMonics:RUNCount? / 875
:POWer:HARMonics:STANDard / 876
:POWer:HARMonics:STATus? / 877
:POWer:HARMonics:THD? / 878
:POWer:INRush:APPLy / 879
:POWer:INRush:EXIT / 880
:POWer:INRush:NEXT / 881
:POWer:ITYPE / 882
:POWer:MODulation:APPLy / 883
:POWer:MODulation:SOURce / 884
:POWer:MODulation:TYPE / 885
:POWer:ONOff:APPLy / 887
:POWer:ONOff:EXIT / 888
:POWer:ONOff:NEXT / 889
:POWer:ONOff:TEST / 890

:POWer:ONOff:THResholds / 891
:POWer:PSRR? / 893
:POWer:PSRR:APPLy / 894
:POWer:PSRR:DATA? / 895
:POWer:PSRR:FREQuency:MAXimum / 896
:POWer:PSRR:FREQuency:MINimum / 897
:POWer:PSRR:FREQuency:MODE / 898
:POWer:PSRR:FREQuency:SINGLE / 899
:POWer:PSRR:PPDecade / 900
:POWer:PSRR:SOURce:INPut / 901
:POWer:PSRR:SOURce:OUTPut / 902
:POWer:PSRR:TRACe / 903
:POWer:PSRR:WGEN:LOAD / 904
:POWer:PSRR:WGEN:VOLTage / 905
:POWer:PSRR:WGEN:VOLTage:PROFile / 906
:POWer:QUALity:APPLy / 907
:POWer:RIPPlE:APPLy / 908
:POWer:SIGNals:AUTosetup / 909
:POWer:SIGNals:CYCLes:HARMonics / 910
:POWer:SIGNals:CYCLes:QUALity / 911
:POWer:SIGNals:DURation:EFFiciency / 912
:POWer:SIGNals:DURation:MODulation / 913
:POWer:SIGNals:DURation:ONOff:OFF / 914
:POWer:SIGNals:DURation:ONOff:ON / 915
:POWer:SIGNals:DURation:RIPPlE / 916
:POWer:SIGNals:DURation:TRANsient / 917
:POWer:SIGNals:IEXPeCted / 918
:POWer:SIGNals:OVERshoot / 919
:POWer:SIGNals:VMAXimum:INRush / 920
:POWer:SIGNals:VMAXimum:ONOff:OFF / 921
:POWer:SIGNals:VMAXimum:ONOff:ON / 922
:POWer:SIGNals:VSTeady:ONOff:OFF / 923
:POWer:SIGNals:VSTeady:ONOff:ON / 924
:POWer:SIGNals:VSTeady:TRANsient / 925
:POWer:SIGNals:SOURce:CURREnt<i> / 926
:POWer:SIGNals:SOURce:VOLTage<i> / 927
:POWer:SLEW:APPLy / 928
:POWer:SLEW:SOURce / 929
:POWer:SWITCh:APPLy / 930
:POWer:SWITCh:CONDUction / 931
:POWer:SWITCh:IREFerence / 932
:POWer:SWITCh:RDS / 933

:POWer:SWITCh:VCE / 934
 :POWer:SWITCh:VREference / 935
 :POWer:TRANsient:APPLy / 936
 :POWer:TRANsient:EXIT / 937
 :POWer:TRANsient:IInitial / 938
 :POWer:TRANsient:INEW / 939
 :POWer:TRANsient:NEXT / 940

33 :RECall Commands

:RECall:ARBitrary[:START] / 944
 :RECall:DBC[:START] / 945
 :RECall:FIlename / 946
 :RECall:LDF[:START] / 947
 :RECall:MASK[:START] / 948
 :RECall:PWD / 949
 :RECall:SETup[:START] / 950
 :RECall:WMEMory<r>[:START] / 951

34 :SAVE Commands

:SAVE:ARBitrary[:START] / 957
 :SAVE:FIlename / 958
 :SAVE:IMAGe[:START] / 959
 :SAVE:IMAGe:FACTors / 960
 :SAVE:IMAGe:FORMat / 961
 :SAVE:LISTer[:START] / 962
 :SAVE:MASK[:START] / 963
 :SAVE:MULTi[:START] / 964
 :SAVE:POWer[:START] / 965
 :SAVE:PWD / 966
 :SAVE:RESults[:START] / 967
 :SAVE:RESults:FORMat:CURSor / 968
 :SAVE:RESults:FORMat:HISTogram / 969
 :SAVE:RESults:FORMat:MASK / 970
 :SAVE:RESults:FORMat:MEASurement / 971
 :SAVE:RESults:FORMat:SEARch / 972
 :SAVE:RESults:FORMat:SEGmented / 973
 :SAVE:SETup[:START] / 974
 :SAVE:WAVEform[:START] / 975
 :SAVE:WAVEform:FORMat / 976
 :SAVE:WAVEform:LENGth / 977
 :SAVE:WAVEform:LENGth:MAX / 978

:SAVE:WAVEform:SEGmented / 979
:SAVE:WMEMemory:SOURce / 980
:SAVE:WMEMemory[:START] / 981

35 :SBUS<n> Commands

General :SBUS<n> Commands / 985
:SBUS<n>:DISPlay / 986
:SBUS<n>:MODE / 987
:SBUS<n>:A429 Commands / 988
:SBUS<n>:A429:AUTosetup / 990
:SBUS<n>:A429:BASE / 991
:SBUS<n>:A429:BAUDrate / 992
:SBUS<n>:A429:FORMat / 993
:SBUS<n>:A429:SIGNal / 994
:SBUS<n>:A429:SOURce / 995
:SBUS<n>:A429:SPEed / 996
:SBUS<n>:A429:TRIGger:LABel / 997
:SBUS<n>:A429:TRIGger:PATtern:DATA / 998
:SBUS<n>:A429:TRIGger:PATtern:SDI / 999
:SBUS<n>:A429:TRIGger:PATtern:SSM / 1000
:SBUS<n>:A429:TRIGger:RANGe / 1001
:SBUS<n>:A429:TRIGger:TYPE / 1002
:SBUS<n>:CAN Commands / 1004
:SBUS<n>:CAN:DISPlay / 1007
:SBUS<n>:CAN:FDSPoint / 1008
:SBUS<n>:CAN:SAMPlepoint / 1009
:SBUS<n>:CAN:SIGNal:BAUDrate / 1010
:SBUS<n>:CAN:SIGNal:DEFinition / 1011
:SBUS<n>:CAN:SIGNal:FDBaudrate / 1012
:SBUS<n>:CAN:SIGNal:XLBaudrate / 1013
:SBUS<n>:CAN:SOURce / 1014
:SBUS<n>:CAN:TRIGger / 1015
:SBUS<n>:CAN:TRIGger:IDFilter / 1018
:SBUS<n>:CAN:TRIGger:PATtern:DATA / 1019
:SBUS<n>:CAN:TRIGger:PATtern:DATA:DLC / 1020
:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth / 1021
:SBUS<n>:CAN:TRIGger:PATtern:DATA:START / 1022
:SBUS<n>:CAN:TRIGger:PATtern:ID / 1023
:SBUS<n>:CAN:TRIGger:PATtern:ID:MODE / 1024
:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge / 1025

- :SBUS<n>:CAN:TRIGger:SYMBolic:SIGNal / 1026
- :SBUS<n>:CAN:TRIGger:SYMBolic:VALue / 1027
- :SBUS<n>:CAN:TRIGger:TYPE / 1028
- :SBUS<n>:CAN:TYPE / 1029
- :SBUS<n>:CAN:XLSPoint / 1030
- :SBUS<n>:IIC Commands / 1031
 - :SBUS<n>:IIC:ASIZe / 1032
 - :SBUS<n>:IIC[:SOURce]:CLOCK / 1033
 - :SBUS<n>:IIC[:SOURce]:DATA / 1034
 - :SBUS<n>:IIC:TRIGger:PATtern:ADDReSS / 1035
 - :SBUS<n>:IIC:TRIGger:PATtern:DATA / 1036
 - :SBUS<n>:IIC:TRIGger:PATtern:DATA2 / 1037
 - :SBUS<n>:IIC:TRIGger:QUALifier / 1038
 - :SBUS<n>:IIC:TRIGger[:TYPE] / 1039
- :SBUS<n>:LIN Commands / 1041
 - :SBUS<n>:LIN:DISPlay / 1043
 - :SBUS<n>:LIN:PARity / 1044
 - :SBUS<n>:LIN:SAMPlepoint / 1045
 - :SBUS<n>:LIN:SIGNal:BAUDrate / 1046
 - :SBUS<n>:LIN:SOURce / 1047
 - :SBUS<n>:LIN:STANDard / 1048
 - :SBUS<n>:LIN:SYNCbreak / 1049
 - :SBUS<n>:LIN:TRIGger / 1050
 - :SBUS<n>:LIN:TRIGger:ID / 1051
 - :SBUS<n>:LIN:TRIGger:PATtern:DATA / 1052
 - :SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth / 1054
 - :SBUS<n>:LIN:TRIGger:PATtern:FORMat / 1055
 - :SBUS<n>:LIN:TRIGger:SYMBolic:FRAME / 1056
 - :SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal / 1057
 - :SBUS<n>:LIN:TRIGger:SYMBolic:VALue / 1058
- :SBUS<n>:M1553 Commands / 1059
 - :SBUS<n>:M1553:AUTOsetup / 1060
 - :SBUS<n>:M1553:BASE / 1061
 - :SBUS<n>:M1553:SOURce / 1062
 - :SBUS<n>:M1553:TRIGger:PATtern:DATA / 1063
 - :SBUS<n>:M1553:TRIGger:RTA / 1064
 - :SBUS<n>:M1553:TRIGger:TYPE / 1065
- :SBUS<n>:SENT Commands / 1066
 - :SBUS<n>:SENT:CLOCK / 1069
 - :SBUS<n>:SENT:CRC / 1070

```

:SBUS<n>:SENT:DISPlay / 1071
:SBUS<n>:SENT:FORMat / 1073
:SBUS<n>:SENT:IDLE / 1075
:SBUS<n>:SENT:LENGth / 1076
:SBUS<n>:SENT:PPULse / 1077
:SBUS<n>:SENT:SIGNal<s>:DISPlay / 1079
:SBUS<n>:SENT:SIGNal<s>:LENGth / 1080
:SBUS<n>:SENT:SIGNal<s>:MULTIplier / 1082
:SBUS<n>:SENT:SIGNal<s>:OFFSet / 1084
:SBUS<n>:SENT:SIGNal<s>:ORDer / 1086
:SBUS<n>:SENT:SIGNal<s>:STARt / 1088
:SBUS<n>:SENT:SOURce / 1090
:SBUS<n>:SENT:TOLerance / 1092
:SBUS<n>:SENT:TRIGger / 1093
:SBUS<n>:SENT:TRIGger:FAST:DATA / 1095
:SBUS<n>:SENT:TRIGger:SLOW:DATA / 1096
:SBUS<n>:SENT:TRIGger:SLOW:ID / 1098
:SBUS<n>:SENT:TRIGger:SLOW:ILENGth / 1100
:SBUS<n>:SENT:TRIGger:TOLerance / 1101

:SBUS<n>:SPI Commands / 1102
:SBUS<n>:SPI:BITOrder / 1104
:SBUS<n>:SPI:CLOCK:SLOPe / 1105
:SBUS<n>:SPI:CLOCK:TIMEout / 1106
:SBUS<n>:SPI:DELay / 1107
:SBUS<n>:SPI:FRAMing / 1108
:SBUS<n>:SPI:SOURce:CLOCK / 1109
:SBUS<n>:SPI:SOURce:FRAME / 1110
:SBUS<n>:SPI:SOURce:MISO / 1111
:SBUS<n>:SPI:SOURce:MOSI / 1112
:SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA / 1113
:SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh / 1114
:SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA / 1115
:SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh / 1116
:SBUS<n>:SPI:TRIGger:TYPE / 1117
:SBUS<n>:SPI:WIDTh / 1118

:SBUS<n>:UART Commands / 1119
:SBUS<n>:UART:BASE / 1121
:SBUS<n>:UART:BAUDrate / 1122
:SBUS<n>:UART:BITOrder / 1123
:SBUS<n>:UART:FRAMing / 1124
:SBUS<n>:UART:PARity / 1125

```

```

:SBUS<n>:UART:POLarity / 1126
:SBUS<n>:UART:SOURce:RX / 1127
:SBUS<n>:UART:SOURce:TX / 1128
:SBUS<n>:UART:TRIGger:BASE / 1129
:SBUS<n>:UART:TRIGger:BURSt / 1130
:SBUS<n>:UART:TRIGger:DATA / 1131
:SBUS<n>:UART:TRIGger:IDLE / 1132
:SBUS<n>:UART:TRIGger:QUALifier / 1133
:SBUS<n>:UART:TRIGger:TYPE / 1134
:SBUS<n>:UART:WIDTh / 1135
:SBUS<n>:USB Commands / 1136
:SBUS<n>:USB:BASE / 1139
:SBUS<n>:USB:SOURce:DMINus / 1140
:SBUS<n>:USB:SOURce:DPLus / 1141
:SBUS<n>:USB:SOURce:DIFFerential / 1142
:SBUS<n>:USB:SPEed / 1143
:SBUS<n>:USB:TRIGger / 1144
:SBUS<n>:USB:TRIGger:ADDResS / 1145
:SBUS<n>:USB:TRIGger:CRC / 1146
:SBUS<n>:USB:TRIGger:DATA / 1147
:SBUS<n>:USB:TRIGger:DATA:LENGth / 1148
:SBUS<n>:USB:TRIGger:ENDPoint / 1149
:SBUS<n>:USB:TRIGger:ET / 1150
:SBUS<n>:USB:TRIGger:FRAMe / 1151
:SBUS<n>:USB:TRIGger:HADDress / 1152
:SBUS<n>:USB:TRIGger:PCHeck / 1153
:SBUS<n>:USB:TRIGger:PID:DATA / 1154
:SBUS<n>:USB:TRIGger:PID:HANDshake / 1155
:SBUS<n>:USB:TRIGger:PID:SPECial / 1156
:SBUS<n>:USB:TRIGger:PID:TOKen / 1157
:SBUS<n>:USB:TRIGger:PORT / 1158
:SBUS<n>:USB:TRIGger:SC / 1159
:SBUS<n>:USB:TRIGger:SEU / 1160
:SBUS<n>:USBPd Commands / 1161
:SBUS<n>:USBPd:SOURce / 1162
:SBUS<n>:USBPd:TRIGger / 1163
:SBUS<n>:USBPd:TRIGger:HEADer / 1164
:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage / 1166
:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage / 1168
:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage / 1169
:SBUS<n>:USBPd:TRIGger:HEADer:VALue / 1171

```

:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier / 1172

36 :SEARch Commands

General :SEARch Commands / 1174

:SEARch:COUNT? / 1175

:SEARch:EVENT / 1176

:SEARch:MODE / 1177

:SEARch:STATe / 1178

:SEARch:EDGE Commands / 1179

:SEARch:EDGE:SLOPe / 1180

:SEARch:EDGE:SOURce / 1181

:SEARch:GLITch Commands / 1182

:SEARch:GLITch:GREaterthan / 1183

:SEARch:GLITch:LESSthan / 1184

:SEARch:GLITch:POLarity / 1185

:SEARch:GLITch:QUALifier / 1186

:SEARch:GLITch:RANGe / 1187

:SEARch:GLITch:SOURce / 1188

:SEARch:PEAK Commands / 1189

:SEARch:PEAK:EXCursion / 1190

:SEARch:PEAK:NPEaks / 1191

:SEARch:PEAK:SOURce / 1192

:SEARch:PEAK:THReshold / 1193

:SEARch:RUNT Commands / 1194

:SEARch:RUNT:POLarity / 1195

:SEARch:RUNT:QUALifier / 1196

:SEARch:RUNT:SOURce / 1197

:SEARch:RUNT:TIME / 1198

:SEARch:TRANSition Commands / 1199

:SEARch:TRANSition:QUALifier / 1200

:SEARch:TRANSition:SLOPe / 1201

:SEARch:TRANSition:SOURce / 1202

:SEARch:TRANSition:TIME / 1203

:SEARch:SERial:A429 Commands / 1204

:SEARch:SERial:A429:LABel / 1205

:SEARch:SERial:A429:MODE / 1206

:SEARch:SERial:A429:PATtern:DATA / 1207

:SEARch:SERial:A429:PATtern:SDI / 1208

:SEARch:SERial:A429:PATtern:SSM / 1209

- :SEARch:SERIal:CAN Commands / 1210
 - :SEARch:SERIal:CAN:MODE / 1211
 - :SEARch:SERIal:CAN:PATtern:DATA / 1213
 - :SEARch:SERIal:CAN:PATtern:DATA:LENGth / 1214
 - :SEARch:SERIal:CAN:PATtern:ID / 1215
 - :SEARch:SERIal:CAN:PATtern:ID:MODE / 1216
 - :SEARch:SERIal:CAN:SYMBolic:MESSage / 1217
 - :SEARch:SERIal:CAN:SYMBolic:SIGNal / 1218
 - :SEARch:SERIal:CAN:SYMBolic:VALue / 1219
- :SEARch:SERIal:IIC Commands / 1220
 - :SEARch:SERIal:IIC:MODE / 1221
 - :SEARch:SERIal:IIC:PATtern:ADDResS / 1223
 - :SEARch:SERIal:IIC:PATtern:DATA / 1224
 - :SEARch:SERIal:IIC:PATtern:DATA2 / 1225
 - :SEARch:SERIal:IIC:QUALifier / 1226
- :SEARch:SERIal:LIN Commands / 1227
 - :SEARch:SERIal:LIN:ID / 1228
 - :SEARch:SERIal:LIN:MODE / 1229
 - :SEARch:SERIal:LIN:PATtern:DATA / 1230
 - :SEARch:SERIal:LIN:PATtern:DATA:LENGth / 1231
 - :SEARch:SERIal:LIN:PATtern:FORMat / 1232
 - :SEARch:SERIal:LIN:SYMBolic:FRAMe / 1233
 - :SEARch:SERIal:LIN:SYMBolic:SIGNal / 1234
 - :SEARch:SERIal:LIN:SYMBolic:VALue / 1235
- :SEARch:SERIal:M1553 Commands / 1236
 - :SEARch:SERIal:M1553:MODE / 1237
 - :SEARch:SERIal:M1553:PATtern:DATA / 1238
 - :SEARch:SERIal:M1553:RTA / 1239
- :SEARch:SERIal:SENT Commands / 1240
 - :SEARch:SERIal:SENT:FAST:DATA / 1241
 - :SEARch:SERIal:SENT:MODE / 1242
 - :SEARch:SERIal:SENT:SLOW:DATA / 1243
 - :SEARch:SERIal:SENT:SLOW:ID / 1244
- :SEARch:SERIal:SPI Commands / 1245
 - :SEARch:SERIal:SPI:MODE / 1246
 - :SEARch:SERIal:SPI:PATtern:DATA / 1247
 - :SEARch:SERIal:SPI:PATtern:WIDTh / 1248
- :SEARch:SERIal:UART Commands / 1249
 - :SEARch:SERIal:UART:DATA / 1250

- :SEARch:SERIal:UART:MODE / 1251
- :SEARch:SERIal:UART:QUALifier / 1253
- :SEARch:SERIal:USB Commands / 1254
 - :SEARch:SERIal:USB:MODE / 1256
 - :SEARch:SERIal:USB:ADDResS / 1257
 - :SEARch:SERIal:USB:CRC / 1258
 - :SEARch:SERIal:USB:DATA / 1259
 - :SEARch:SERIal:USB:DATA:LENGth / 1260
 - :SEARch:SERIal:USB:ENDPoint / 1261
 - :SEARch:SERIal:USB:ET / 1262
 - :SEARch:SERIal:USB:FRAMe / 1263
 - :SEARch:SERIal:USB:HADDress / 1264
 - :SEARch:SERIal:USB:PID:DATA / 1265
 - :SEARch:SERIal:USB:PID:HANDshake / 1266
 - :SEARch:SERIal:USB:PID:SPECial / 1267
 - :SEARch:SERIal:USB:PID:TOKen / 1268
 - :SEARch:SERIal:USB:PORT / 1269
 - :SEARch:SERIal:USB:SC / 1270
 - :SEARch:SERIal:USB:SEU / 1271
- :SEARch:SERIal:USBPd Commands / 1272
 - :SEARch:SERIal:USBPd:MODE / 1273
 - :SEARch:SERIal:USBPd:HEADer / 1274
 - :SEARch:SERIal:USBPd:HEADer:CMESsage / 1276
 - :SEARch:SERIal:USBPd:HEADer:DMESsage / 1278
 - :SEARch:SERIal:USBPd:HEADer:EMESsage / 1279
 - :SEARch:SERIal:USBPd:HEADer:QUALifier / 1281
 - :SEARch:SERIal:USBPd:HEADer:VALue / 1282

37 :STATus Commands

- General :STATus Commands / 1286
 - :STATus:PRESet / 1287
- :STATus:OPERation Commands / 1288
 - :STATus:OPERation:BIT:CONDition? / 1293
 - :STATus:OPERation:BIT:ENABle / 1294
 - :STATus:OPERation:BIT:NTRansition / 1295
 - :STATus:OPERation:BIT:PTRansition / 1296
 - :STATus:OPERation:BIT[:EVENT]? / 1297
 - :STATus:OPERation:CONDition? / 1298
 - :STATus:OPERation:ENABle / 1299
 - :STATus:OPERation:NTRansition / 1300

```

:STATus:OPERation:PTRansition / 1301
:STATus:OPERation[:EVENT]? / 1302

:STATus:OPERation:ARM Commands / 1303
:STATus:OPERation:ARM:BIT<b>:CONDition? / 1306
:STATus:OPERation:ARM:BIT<b>:ENABle / 1307
:STATus:OPERation:ARM:BIT<b>:NTRansition / 1308
:STATus:OPERation:ARM:BIT<b>:PTRansition / 1309
:STATus:OPERation:ARM:BIT<b>[:EVENT]? / 1310
:STATus:OPERation:ARM:CONDition? / 1311
:STATus:OPERation:ARM:ENABle / 1312
:STATus:OPERation:ARM:NTRansition / 1313
:STATus:OPERation:ARM:PTRansition / 1314
:STATus:OPERation:ARM[:EVENT]? / 1315

:STATus:OPERation:HARDware Commands / 1316
:STATus:OPERation:HARDware:BIT<b>:CONDition? / 1319
:STATus:OPERation:HARDware:BIT<b>:ENABle / 1320
:STATus:OPERation:HARDware:BIT<b>:NTRansition / 1321
:STATus:OPERation:HARDware:BIT<b>:PTRansition / 1322
:STATus:OPERation:HARDware:BIT<b>[:EVENT]? / 1323
:STATus:OPERation:HARDware:CONDition? / 1324
:STATus:OPERation:HARDware:ENABle / 1325
:STATus:OPERation:HARDware:NTRansition / 1326
:STATus:OPERation:HARDware:PTRansition / 1327
:STATus:OPERation:HARDware[:EVENT]? / 1328

:STATus:OPERation:LOCal Commands / 1329
:STATus:OPERation:LOCal:BIT<b>:CONDition? / 1332
:STATus:OPERation:LOCal:BIT<b>:ENABle / 1333
:STATus:OPERation:LOCal:BIT<b>:NTRansition / 1334
:STATus:OPERation:LOCal:BIT<b>:PTRansition / 1335
:STATus:OPERation:LOCal:BIT<b>[:EVENT]? / 1336
:STATus:OPERation:LOCal:CONDition? / 1337
:STATus:OPERation:LOCal:ENABle / 1338
:STATus:OPERation:LOCal:NTRansition / 1339
:STATus:OPERation:LOCal:PTRansition / 1340
:STATus:OPERation:LOCal[:EVENT]? / 1341

:STATus:OPERation:MTESt Commands / 1342
:STATus:OPERation:MTESt:BIT<b>:CONDition? / 1345
:STATus:OPERation:MTESt:BIT<b>:ENABle / 1346
:STATus:OPERation:MTESt:BIT<b>:NTRansition / 1347
:STATus:OPERation:MTESt:BIT<b>:PTRansition / 1348

```

```

:STATus:OPERation:MTEST:BIT<b>[:EVENT]? / 1349
:STATus:OPERation:MTEST:CONDition? / 1350
:STATus:OPERation:MTEST:ENABLE / 1351
:STATus:OPERation:MTEST:NTRansition / 1352
:STATus:OPERation:MTEST:PTRansition / 1353
:STATus:OPERation:MTEST[:EVENT]? / 1354

:STATus:OPERation:OVERload Commands / 1355
:STATus:OPERation:OVERload:BIT<b>:CONDition? / 1358
:STATus:OPERation:OVERload:BIT<b>:ENABLE / 1359
:STATus:OPERation:OVERload:BIT<b>:NTRansition / 1360
:STATus:OPERation:OVERload:BIT<b>:PTRansition / 1361
:STATus:OPERation:OVERload:BIT<b>[:EVENT]? / 1362
:STATus:OPERation:OVERload:CONDition? / 1363
:STATus:OPERation:OVERload:ENABLE / 1364
:STATus:OPERation:OVERload:NTRansition / 1365
:STATus:OPERation:OVERload:PTRansition / 1366
:STATus:OPERation:OVERload[:EVENT]? / 1367

:STATus:OPERation:OVERload:PFault Commands / 1368
:STATus:OPERation:OVERload:PFault:BIT<b>:CONDition? / 1371
:STATus:OPERation:OVERload:PFault:BIT<b>:ENABLE / 1372
:STATus:OPERation:OVERload:PFault:BIT<b>:NTRansition / 1373
:STATus:OPERation:OVERload:PFault:BIT<b>:PTRansition / 1374
:STATus:OPERation:OVERload:PFault:BIT<b>[:EVENT]? / 1375
:STATus:OPERation:OVERload:PFault:CONDition? / 1376
:STATus:OPERation:OVERload:PFault:ENABLE / 1377
:STATus:OPERation:OVERload:PFault:NTRansition / 1378
:STATus:OPERation:OVERload:PFault:PTRansition / 1379
:STATus:OPERation:OVERload:PFault[:EVENT]? / 1380

:STATus:OPERation:POWer Commands / 1381
:STATus:OPERation:POWer:BIT<b>:CONDition? / 1384
:STATus:OPERation:POWer:BIT<b>:ENABLE / 1385
:STATus:OPERation:POWer:BIT<b>:NTRansition / 1386
:STATus:OPERation:POWer:BIT<b>:PTRansition / 1387
:STATus:OPERation:POWer:BIT<b>[:EVENT]? / 1388
:STATus:OPERation:POWer:CONDition? / 1389
:STATus:OPERation:POWer:ENABLE / 1390
:STATus:OPERation:POWer:NTRansition / 1391
:STATus:OPERation:POWer:PTRansition / 1392
:STATus:OPERation:POWer[:EVENT]? / 1393

:STATus:TRIGger Commands / 1394

```

```

:STATus:TRIGger:BIT<b>:CONDition? / 1397
:STATus:TRIGger:BIT<b>:ENABle / 1398
:STATus:TRIGger:BIT<b>:NTRansition / 1399
:STATus:TRIGger:BIT<b>:PTRansition / 1400
:STATus:TRIGger:BIT<b>[:EVENT]? / 1401
:STATus:TRIGger:CONDition? / 1402
:STATus:TRIGger:ENABle / 1403
:STATus:TRIGger:NTRansition / 1404
:STATus:TRIGger:PTRansition / 1405
:STATus:TRIGger[:EVENT]? / 1406
:STATus:USER Commands / 1407
:STATus:USER:BIT<b>:CONDition? / 1410
:STATus:USER:BIT<b>:ENABle / 1411
:STATus:USER:BIT<b>:NTRansition / 1412
:STATus:USER:BIT<b>:PTRansition / 1413
:STATus:USER:BIT<b>[:EVENT]? / 1414
:STATus:USER:CONDition? / 1415
:STATus:USER:ENABle / 1416
:STATus:USER:NTRansition / 1417
:STATus:USER:PTRansition / 1418
:STATus:USER[:EVENT]? / 1419

```

38 :SYSTem Commands

```

:SYSTem:DATE / 1425
:SYSTem:DIIdentifier? / 1426
:SYSTem:DSP / 1427
:SYSTem:ERRor:ALL? / 1428
:SYSTem:ERRor:CODE[:NEXT]? / 1429
:SYSTem:ERRor:COUNt? / 1430
:SYSTem:ERRor:EXTended? / 1431
:SYSTem:ERRor[:NEXT]? / 1432
:SYSTem:HELP:HEADers? / 1434
:SYSTem:HID? / 1435
:SYSTem:LOCK / 1436
:SYSTem:LOCK:OWNer? / 1437
:SYSTem:LOCK:RELease / 1438
:SYSTem:LOCK:REQuest? / 1439
:SYSTem:PERSon[:MANufacturer] / 1440
:SYSTem:PERSon[:MANufacturer]:DEFault / 1441
:SYSTem:POWercycle / 1442
:SYSTem:PRECision:LENGth / 1443

```

- :SYSTem:PRECision:PEAK / 1444
- :SYSTem:PRESet / 1445
- :SYSTem:PRODUCTION:FRESet / 1448
- :SYSTem:PROTection:LOCK / 1449
- :SYSTem:RLOGger / 1450
- :SYSTem:RLOGger:DESTination / 1451
- :SYSTem:RLOGger:DISPlay / 1452
- :SYSTem:RLOGger:FNAME / 1453
- :SYSTem:RLOGger:STATe / 1454
- :SYSTem:RLOGger:TRANSPARENT / 1455
- :SYSTem:RLOGger:WMODE / 1456
- :SYSTem:SETup / 1457
- :SYSTem:SHUTdown / 1459
- :SYSTem:TIME / 1460
- :SYSTem:TIME:DSAVing? / 1461
- :SYSTem:TIME:NTPRotocol / 1462
- :SYSTem:TIME:ZONE / 1463
- :SYSTem:TOUCH / 1464
- :SYSTem:VERSion? / 1465

39 :TIMebase Commands

- :TIMebase[:MAIN]:POSition / 1469
- :TIMebase[:MAIN]:RANGe / 1470
- :TIMebase[:MAIN]:SCALE / 1471
- :TIMebase:MODE / 1472
- :TIMebase:REFerence / 1473
- :TIMebase:REFerence:LOCation / 1474
- :TIMebase:VERNier / 1475
- :TIMebase:WINDow:POSition / 1476
- :TIMebase:WINDow:RANGe / 1477
- :TIMebase:WINDow:SCALE / 1478

40 :TRIGger Commands

- General :TRIGger Commands / 1481
- :TRIGger:FORCe / 1483
- :TRIGger:HFReject / 1484
- :TRIGger:HOLDoff / 1485
- :TRIGger:HOLDoff:MAXimum / 1486
- :TRIGger:HOLDoff:MINimum / 1487
- :TRIGger:HOLDoff:RANDom / 1488
- :TRIGger:HYSTeresis:STATe / 1489

- :TRIGger:JFRee / 1490
- :TRIGger:LEVel:ASeTup / 1491
- :TRIGger:LEVel:HIGH / 1492
- :TRIGger:LEVel:LOW / 1493
- :TRIGger:MODE / 1494
- :TRIGger:NREJect / 1495
- :TRIGger:SWEEP / 1496
- :TRIGger:DElay Commands / 1497
 - :TRIGger:DElay:ARM:SLOPe / 1499
 - :TRIGger:DElay:ARM:SOURce / 1500
 - :TRIGger:DElay:TDElay:TMAX / 1501
 - :TRIGger:DElay:TDElay:TMIN / 1502
 - :TRIGger:DElay:TRIGger:SLOPe / 1503
 - :TRIGger:DElay:TRIGger:SOURce / 1504
- :TRIGger[:EDGE] Commands / 1505
 - :TRIGger[:EDGE]:COUPling / 1506
 - :TRIGger[:EDGE]:LEVel / 1507
 - :TRIGger[:EDGE]:REJect / 1508
 - :TRIGger[:EDGE]:SLOPe / 1509
 - :TRIGger[:EDGE]:SOURce / 1510
- :TRIGger:GLITCh Commands / 1511
 - :TRIGger:GLITCh:GREaterthan / 1513
 - :TRIGger:GLITCh:LESSthan / 1514
 - :TRIGger:GLITCh:LEVel / 1515
 - :TRIGger:GLITCh:POLarity / 1516
 - :TRIGger:GLITCh:QUALifier / 1517
 - :TRIGger:GLITCh:RANGe / 1518
 - :TRIGger:GLITCh:SOURce / 1519
- :TRIGger:OR Commands / 1520
 - :TRIGger:OR / 1521
- :TRIGger:PATtern Commands / 1522
 - :TRIGger:PATtern / 1523
 - :TRIGger:PATtern:FORMat / 1525
 - :TRIGger:PATtern:GREaterthan / 1526
 - :TRIGger:PATtern:LESSthan / 1527
 - :TRIGger:PATtern:QUALifier / 1528
 - :TRIGger:PATtern:RANGe / 1529
- :TRIGger:RUNT Commands / 1530
 - :TRIGger:RUNT:POLarity / 1531

- :TRIGger:RUNT:QUALifier / 1532
- :TRIGger:RUNT:SOURce / 1533
- :TRIGger:RUNT:TIME / 1534
- :TRIGger:SHOLd Commands / 1535
 - :TRIGger:SHOLd:SLOPe / 1536
 - :TRIGger:SHOLd:SOURce:CLOCK / 1537
 - :TRIGger:SHOLd:SOURce:DATA / 1538
 - :TRIGger:SHOLd:TIME:HOLD / 1539
 - :TRIGger:SHOLd:TIME:SETup / 1540
- :TRIGger:TRANSition Commands / 1541
 - :TRIGger:TRANSition:QUALifier / 1542
 - :TRIGger:TRANSition:SLOPe / 1543
 - :TRIGger:TRANSition:SOURce / 1544
 - :TRIGger:TRANSition:TIME / 1545
- :TRIGger:ZONE Commands / 1546
 - :TRIGger:ZONE<z>:LOGic / 1547
 - :TRIGger:ZONE<z>:MODE / 1548
 - :TRIGger:ZONE<z>:PLACement / 1549
 - :TRIGger:ZONE<z>:SOURce / 1550
 - :TRIGger:ZONE<z>:STATe / 1551
 - :TRIGger:ZONE<z>:VALidity? / 1552

41 :WAVeform Commands

- :WAVeform:BYTeorder / 1561
- :WAVeform:COUNt? / 1562
- :WAVeform:DATA? / 1563
- :WAVeform:FORMat / 1565
- :WAVeform:POINts / 1566
- :WAVeform:POINts:MODE / 1568
- :WAVeform:PREamble? / 1570
- :WAVeform:SEGMENTed:ALL / 1573
- :WAVeform:SEGMENTed:COUNt? / 1574
- :WAVeform:SEGMENTed:TTAG? / 1575
- :WAVeform:SEGMENTed:XLIST? / 1576
- :WAVeform:SOURce / 1577
- :WAVeform:SOURce:SUBSource / 1581
- :WAVeform:TYPE? / 1582
- :WAVeform:UNSigned / 1583
- :WAVeform:VIEW / 1584
- :WAVeform:XINCrement? / 1585

:WAVeform:XORigin? / 1586
:WAVeform:XREFerence? / 1587
:WAVeform:YINCrement? / 1588
:WAVeform:YORigin? / 1589
:WAVeform:YREFerence? / 1590

42 :WGEN<w> Commands

:WGEN<w>:ARBitrary:BYTeorder / 1596
:WGEN<w>:ARBitrary:DATA / 1597
:WGEN<w>:ARBitrary:DATA:ATTRibute:POINts? / 1600
:WGEN<w>:ARBitrary:DATA:CLEar / 1601
:WGEN<w>:ARBitrary:DATA:DAC / 1602
:WGEN<w>:ARBitrary:INTErpolate / 1603
:WGEN<w>:ARBitrary:STORe / 1604
:WGEN<w>:DCMode / 1605
:WGEN<w>:FREQuency / 1606
:WGEN<w>:FUNCTion / 1607
:WGEN<w>:FUNCTion:PULSe:WIDTh / 1613
:WGEN<w>:FUNCTion:RAMP:SYMMetry / 1614
:WGEN<w>:FUNCTion:SQUare:DCYCLE / 1615
:WGEN<w>:MODulation:AM:DEPTTh / 1616
:WGEN<w>:MODulation:AM:FREQuency / 1617
:WGEN<w>:MODulation:FM:DEViation / 1618
:WGEN<w>:MODulation:FM:FREQuency / 1619
:WGEN<w>:MODulation:FUNCTion / 1620
:WGEN<w>:MODulation:NOISe / 1621
:WGEN<w>:MODulation:STATe / 1622
:WGEN<w>:MODulation:TYPE / 1623
:WGEN<w>:OUTPut / 1624
:WGEN<w>:OUTPut:BURSt / 1625
:WGEN<w>:OUTPut:BURSt:COUNt / 1626
:WGEN<w>:OUTPut:LOAD / 1627
:WGEN<w>:OUTPut:MODE / 1628
:WGEN<w>:OUTPut:POLarity / 1629
:WGEN<w>:PERiod / 1630
:WGEN<w>:RST / 1631
:WGEN<w>:VOLTage / 1632
:WGEN<w>:VOLTage:HIGH / 1633
:WGEN<w>:VOLTage:LOW / 1634
:WGEN<w>:VOLTage:OFFSet / 1635

43 :WMEemory<r> Commands

- :WMEemory<r>:CLEar / 1639
- :WMEemory<r>:DISPlay / 1640
- :WMEemory<r>:LABel / 1641
- :WMEemory<r>:SAVE / 1642
- :WMEemory<r>:SKEW / 1643
- :WMEemory<r>:YOFFset / 1644
- :WMEemory<r>:YRANge / 1645
- :WMEemory<r>:YSCale / 1646

44 Obsolete and Discontinued Commands

- :CHANnel<n>:BWLimit / 1656
- :MEASure:SCRatch / 1657
- :SBUS<n>:SPI:SOURce:DATA / 1658
- :TIMEbase[:MAIN]:DELay / 1659
- :TRIGger:DELay:TDELay:TIME / 1660

45 Error Messages

46 Status Reporting

- Status Reporting Data Structures / 1672
- Output Queue / 1673
- Message Queue / 1674
- Clearing Registers and Queues / 1675
- Status Reporting Decision Chart / 1676
- Example: Checking for Armed Status / 1677
- Example: Waiting for IO Operation Complete / 1682

47 Synchronizing Acquisitions

- Synchronization in the Programming Flow / 1686
 - Set Up the Oscilloscope / 1686
 - Acquire a Waveform / 1686
 - Retrieve Results / 1686
- Blocking Synchronization / 1687
- Polling Synchronization With Timeout / 1688
- Synchronizing with a Single-Shot Device Under Test (DUT) / 1690
- Synchronization with an Averaging Acquisition / 1693

Example: Blocking and Polling Synchronization / 1695

48 More About Oscilloscope Commands

- Command Classifications / 1708
 - Core Commands / 1708
 - Non-Core Commands / 1708
 - Obsolete Commands / 1708
- Valid Command/Query Strings / 1709
 - Program Message Syntax / 1709
 - Duplicate Mnemonics / 1714
 - Tree Traversal Rules and Multiple Commands / 1714
- Query Return Values / 1716

49 Programming Examples

- VISA COM Examples / 1718
 - VISA COM Example in Visual Basic / 1718
 - VISA COM Example in C# / 1727
 - VISA COM Example in Visual Basic .NET / 1736
 - VISA COM Example in Python / 1745
- VISA Examples / 1752
 - VISA Example in C / 1752
 - VISA Example in C# / 1761
 - VISA Example in Visual Basic .NET / 1772
 - VISA Example in Python / 1782
- VISA.NET Examples / 1789
 - VISA.NET Example in C# / 1789
 - VISA.NET Example in Visual Basic .NET / 1795
 - VISA.NET Example in Python / 1801
- SICL Examples / 1808
 - SICL Example in C / 1808
- SCPI.NET Examples / 1818

Index

1 What's New

What's New in Version 10.21 /	42
What's New in Version 10.20 /	43
What's New in Version 10.15 /	45
What's New in Version 10.10 /	46
What's New in Version 10.07 /	47
What's New in Version 10.06 /	50
What's New in Version 10.05 /	52
Version 10.00 at Introduction /	54
Command Differences From 3000G X-Series Oscilloscopes /	55

What's New in Version 10.21

New features in version 10.21 of the InfiniiVision HD3-Series oscilloscope software are:

- A new option for detecting peaks when decimating the raw acquisition record into the *analysis record*.
- The waveform generator square wave function now offers narrower and wider duty cycles depending on the square wave frequency.

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

New Commands

Command	Description
:DISPlay:DIALog:CLOSe (see page 426)	Closes all open dialog boxes in the graphical user interface.
:DISPlay:DIALog:TRANSPARENT (see page 427)	Enables or disables the Transparent Dialogs user preference option.
:SYSTem:PRECision:PEAK (see page 1444)	Enables or disables the Analyze Peaks option for the analysis record.

Changed Commands

Command	Differences
:WGEN<w>:FUNCTION:SQUare:DCYCLE (see page 1615)	The duty cycle percentage can be from 0.5% to 99.5% depending on the square wave frequency.

What's New in Version 10.20

New features in version 10.20 of the InfiniiVision HD3-Series oscilloscope software are:

- Support for the new HD300QJTA jitter analysis license (including clock recovery).
- USB PD (Power Delivery) serial decode and triggering with the HD300USBA USB analysis license.
- USB 2.0 triggering and serial decode and triggering with the HD300USBA USB analysis license.
- Color grade waveforms.

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

New Commands

Command	Description
:CHANnel<n>:PROBe:EXTeRnal:OFFSet (see page 362)	Sets the external vertical value for the probe that is represented at the center of the display for the selected channel.
:CLOCK Commands (see page 379)	Commands for controlling the clock recovery feature.
:CGRade Commands (see page 333)	Commands for controlling the color grade waveform analysis feature.
:FUNctIon<m>:AUTosetup (see page 524)	Automatically sets the Y scale and Y offset to optimum values for the math function waveform.
:FUNctIon<m>:AUTosetup:ENABle (see page 525)	Lets you enable or disable the automatic setup of a math function's scale and offset.
:JITTer Commands (see page 593)	Commands for controlling the license-enabled jitter analysis feature.
:MEASure:CTCnwidth (see page 679)	Returns the -Width to -Width jitter value.
:MEASure:CTCPwidth (see page 681)	Returns the +Width to +Width jitter value.
:MEASure:NCJitter (see page 708)	Returns the N-Period to N-Period jitter value.
:MEASure:NPERiod (see page 712)	Returns the N-Period jitter value.
:MEASure:TIEData (see page 752)	Returns the Data TIE jitter value.
:MEASure:TIEClock (see page 750)	Returns the Clock TIE jitter value.

Command	Description
:SBUS<n>:USB Commands (see page 1136)	Commands for using the USB 2.0 triggering and serial decode feature.
:SBUS<n>:USBPD Commands (see page 1161)	Commands for using the USB PD triggering and serial decode feature.
:SEARCh:SErIal:USB Commands (see page 1254)	Commands for searching USB 2.0 serial decode events.
:SEARCh:SErIal:USBPD Commands (see page 1272)	Commands for searching USB PD serial decode events.

Changed Commands

Command	Differences
:SBUS<n>:MODE (see page 987)	The USB and USBpd modes are now available with the HD300USBA USB analysis license.

Discontinued Commands

Discontinued Command	Current Command Equivalent	Comments
:CHANnel<n>:PROBe:MMODEl	none	Removed because the HD3-Series oscilloscopes do not support these probe model numbers.

What's New in Version 10.15

New features in version 10.15 of the InfiniiVision HD3-Series oscilloscope software are:

- Support for the new HD300PWRA power measurements and analysis license.
- Support for the new PS000XA probes.

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

New Commands

Command	Description
:CHANnel<n>:PROBe:CALibrati on:VALidity? (see page 359)	PS000XA probes only, this query returns whether the calibration factors for the probe are still valid.
:CHANnel<n>:PROBe:STATus? (see page 370)	PS000XA probes only, this query returns the probe status.
:MEASure Power Commands (see page 769)	:MEASure commands available when the HD300PWRA power measurements and analysis application is licensed and enabled.
:POWer Commands (see page 837)	Commands for the HD300PWRA power measurements and analysis application.
:SAVE:POWer[:START] (see page 965)	Saves the power measurement application's current harmonics analysis results to a file.

Changed Commands

Command	Differences
:CHANnel<n>:PROBe:CALibrati on (see page 358)	This command is now also used with the PS000XA probes.
:CHANnel<n>:PROBe:STYPe (see page 371)	With PS000XA probes, the only valid value is ISOLated.

What's New in Version 10.10

New features in version 10.10 of the InfiniiVision HD3-Series oscilloscope software are:

- ARINC 429 serial decode and triggering option.
- MIL-STD 1553 serial decode and triggering option.
- FFT Max options for enabling/disabling, changing the vertical scale and offset, and resetting the number of waveforms evaluated.

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

New Commands

Command	Description
:FFTMaX:OFFSet (see page 492)	When the FFT Max function is enabled, this command sets its vertical offset (center of grid value).
:FFTMaX:RESet (see page 493)	When the FFT Max function is enabled, this command clears the number of waveforms evaluated.
:FFTMaX:SCALe (see page 494)	When the FFT Max function is enabled, this command sets its vertical scale.
:FFTMaX:STATe (see page 495)	Enables or disables the FFT Max function.
:SBUS<n>:A429 Commands (see page 988)	Commands for using the ARINC 429 protocol decode and triggering decode feature.
:SBUS<n>:M1553 Commands (see page 1059)	Commands for using the MIL-STD-1553 protocol decode and triggering decode feature.
:SEARCh:SERIal:A429 Commands (see page 1204)	Commands for searching ARINC 429 protocol decode events.
:SEARCh:SERIal:M1553 Commands (see page 1236)	Commands for searching MIL-STD-1553 protocol decode events.

Changed Commands

Command	Differences
:SBUS<n>:MODE (see page 987)	The A429 (ARINC 429) and M1553 (MIL-STD-1553) modes are now available with the Aerospace Serial Analysis license.

What's New in Version 10.07

New features in version 10.07 of the InfiniiVision HD3-Series oscilloscope software are:

- The FFT waveform and a math function FFT waveform can now be gated by markers as well as by zoom (which was previously available).
- All math function waveforms can now be gated by main (none), zoom, main and zoom, or markers.
- There are now four reference waveforms (:WMEMory<r>) instead of two.
- Auto versus manual sample rate and memory depth controls are now independent. Before, either both sample rate and memory depth were automatically determined based the time/div setting, or both were manually specified (Digitizer mode).
- The frequency response analysis amplitude profile now has options for all decade ranges supported by the built-in waveform generator.
- "PBUS" (protocol decode bus) is now an alias for "SBUS" (serial bus) because terminology has changed in the HD3-Series oscilloscope. The HD3-Series oscilloscope has a **[Protocol Decode]** front panel key and refers to the feature as "Protocol Decode/Triggering". Previous InfiniiVision X-Series oscilloscopes had a **[Serial]** front panel key and refer to the feature as "Serial Decode/Triggering".
- The minimum baudrate setting for UART protocol decode is now 200 b/s.

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

New Commands

Command	Description
:FUNction<m>:GATE (see page 553)	Specifies the input waveform data on which the math function is performed.
:MEASure:HYSTeresis (see page 707)	Disable or reenale the minimum signal amplitude limit for making measurements.
:TRIGger:DElay:TDElay:TMAX (see page 1501)	Sets the minimum delay time between the arming edge and the trigger edge.
:TRIGger:DElay:TDElay:TMIN (see page 1502)	Sets the minimum delay time between the arming edge and the trigger edge.
:TRIGger:HYSTeresis:STATe (see page 1489)	Disable or reenale the minimum signal amplitude limit for making measurements.

Changed Commands

Command	Differences
:ACQuire:DIGitizer (see page 290)	This command enables or disables <i>both</i> the :ACQuire:SRATe[:ANALog]:AUTO and :ACQuire:POINts[:ANALog]:AUTO settings at the same time (though those settings are now independent).
:ACQuire:MANual (see page 292)	This command enables or disables <i>both</i> the :ACQuire:SRATe[:ANALog]:AUTO and :ACQuire:POINts[:ANALog]:AUTO settings at the same time (though those settings are now independent).
:ACQuire:POINts[:ANALog]:AUTO (see page 298)	This command enables or disables whether the memory depth is automatically determined or is set by the :ACQuire:POINts[:ANALog] command. Previously, this setting had to match the :ACQuire:SRATe[:ANALog]:AUTO setting, but now it is independent.
:ACQuire:SRATe[:ANALog]:AUTO (see page 308)	This command enables or disables whether the sample rate is automatically determined or is set by the :ACQuire:SRATe[:ANALog] command. Previously, this setting had to match the :ACQuire:POINts[:ANALog]:AUTO setting, but now it is independent.
:BLANK (see page 270)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.
:DIGitize (see page 271)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.
:DISPlay:GRATicule:SET (see page 435)	The PBUS<n> parameter is now available as an equivalent to the SBUS<n> parameter.
:FFT:GATE (see page 478)	The MARKers option has been added for performing the FFT on the input waveform time range between the horizontal markers.
:FRANalysis:WGEN:VOLTage (see page 513)	There are now these <range> options to cover the full range of frequencies available with the built-in waveform generator: F10MILLIHZ, F100MILLIHZ, F1HZ, F10HZ, and F100MHZ.
:FUNCTion<m>:FFT:GATE (see page 540)	The MARKers option has been added for performing the FFT on the input waveform time range between the horizontal markers.
:LISTer:DISPlay (see page 615)	The PBUS1 and PBUS2 parameters are now available as equivalents of the SBUS1 and SBUS2 parameters.
:RECall:DBC[:START] (see page 945)	The PBUS<n> parameter is now available as an equivalent to the SBUS<n> parameter.
:RECall:LDF[:START] (see page 947)	The PBUS<n> parameter is now available as an equivalent to the SBUS<n> parameter.
:SBUS<n>:UART:BAUDrate (see page 1122)	The minimum baudrate setting is now 200 b/s.
:STATus? (see page 278)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.

Command	Differences
:SYSTem:TIME:DSAVing? (see page 1461)	This is no longer an option that can be set. For the selected time zone, this query returns whether "automatically adjust clock for Daylight Savings Time" is enabled or disabled.
:SYSTem:TIME:ZONE (see page 1463)	There are now named time zones to select from as well as UTC offsets.
:TRIGger:MODE (see page 1494)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.
:VIEW (see page 281)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.
:WAVEform:SOURce (see page 1577)	The PBUS{1 2} parameters are now available as equivalents of the SBUS{1 2} parameters.

Obsolete Commands

Obsolete Command	Current Command Equivalent	Behavior Differences
:TRIGger:DElay:TDElay:TIME (see page 1660)	:TRIGger:DElay:TDElay:TMIN (see page 1502)	The obsolete command was a delay time before looking for the trigger edge. The current command equivalent specifies the minimum time of a range in which to look for the trigger edge.

Discontinued Commands

Discontinued Command	Current Command Equivalent	Comments
:TRIGger:DElay:TRIGger:COUNt	none	You cannot trigger on an Nth edge.

What's New in Version 10.06

New features in version 10.06 of the InfiniiVision HD3-Series oscilloscope software are:

- Edge Then Edge trigger mode.
- SENT serial decode and triggering option.
- Search for runt pulses.
- New command and query for the anti-alias control.
- New queries for returning the **User** sample rate and memory depth graphical user interface settings.
- New command/query for displaying waveforms with lines connecting the sampled data points or just waveform data points.

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

New Commands

Command	Description
:ACQuire:AALias? (see page 286)	Returns the current state of the oscilloscope acquisition anti-alias control.
:ACQuire:DAALias (see page 289)	Sets the oscilloscope's disable anti-alias mode or queries the setting.
:ACQuire:MANual:POINts? (see page 294)	Returns the User memory depth graphical user interface setting.
:ACQuire:MANual:SRAt? (see page 295)	Returns the User sample rate graphical user interface setting.
:DISPlay:VECTors (see page 450)	Turns vector display on or off.
:DISPlay:XYColor (see page 451)	Specifies the color scheme (and persistence color) used for the XY horizontal time mode waveform.
:FRANalysis:SWEp:POINts (see page 510)	Specifies the number of points in the frequency response analysis.
:SBUS<n>:SENT Commands (see page 1066)	Commands for using the SENT protocol decode and triggering decode feature.
:SEARCh:RUNT Commands (see page 1194)	Commands for searching runt pulses.
:SEARCh:SERial:SENT Commands (see page 1240)	Commands for searching SENT protocol decode events.
:TRIGger:DELAy Commands (see page 1497)	Commands for setting up the Edge Then Edge trigger mode.

Command	Description
:WGEN<w>:OUTPut:BURSt (see page 1625)	When the BURSt output mode is selected, this command causes the burst count number of cycles of the defined waveform to be output.
:WGEN<w>:OUTPut:BURSt:CO UNt (see page 1626)	When the BURSt output mode is selected, this command sets the burst count number.
:WGEN<w>:OUTPut:MODE (see page 1628)	Specifies whether the defined waveform is output continuously or as a specified number of cycles (burst).

Changed Commands

Command	Differences
:FUNctIon<m>:OPERation (see page 561)	The PEAK operation for the Peak-Peak visualization math function is now available.
:SBUS<n>:MODE (see page 987)	The SENT mode is now available with the Automotive Serial Analysis license.
:SEARch:MODE (see page 1177)	The RUNT option has been added to enable searching for runt pulses (see ":SEARch:RUNT Commands" on page 1194).
:TRIGger:MODE (see page 1494)	The DELay option is now available for setting the Edge Then Edge trigger mode. See ":TRIGger:DELay Commands" on page 1497.
:WAVEform:SOURce:SUBSource (see page 1581)	You can use FAST (alias for SUB0) to get SENT fast channel data or SLOW (alias for SBUS1) to get SENT slow channel data.

What's New in Version 10.05

New features in version 10.05 of the InfiniiVision HD3-Series oscilloscope software are:

- New query for the maximum number of points when Manual sampling rates and memory depths are enabled.
- New command for Factory Reset.
- Addition of the XY and ROLL time modes.

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

New Commands

Command	Description
:ACQUIRE:POINTS:MAX? (see page 301)	Returns the oscilloscope's acquisition points maximum limit.
:MODEL? (see page 273)	Returns the oscilloscope's model number.
:SYSTEM:PRODUCTION:FRESet (see page 1448)	Wipes the oscilloscope's non-volatile storage like Secure Erase , but it does not perform the National Industrial Security Program Operation Manual (NISPOM) Chapter 8 procedure and is much faster.

Changed Commands

Command	Differences
:TIMEbase:MODE (see page 1472)	The XY and ROLL modes are now available.

Obsolete Commands

Obsolete Command	Current Command Equivalent	Behavior Differences
:CHANNEL<n>:BWLimit (see page 1656)	:CHANNEL<n>:BANDwidth (see page 349) 20E+06	In earlier versions of the HD3-Series software, the :CHANNEL<n>:BWLimit command mapped to a 40 MHz bandwidth limit on the channel. Now that the "global" :ACQUIRE:BANDwidth command has been replaced by the "per channel" :CHANNEL<n>:BANDwidth command, the :CHANNEL<n>:BWLimit returns to the legacy 20 MHz limit of previous InfiniiVision oscilloscopes.

Discontinued
Commands

Discontinued Command	Current Command Equivalent	Comments
:ACQuire:BANDwidth	:CHANnel<n>:BANDwidth (see page 349)	In earlier versions of the HD3-Series software, the :ACQuire:BANDwidth command was a "global" bandwidth setting. The :CHANnel<n>:BANDwidth command is a "per channel" bandwidth setting (although its 5 MHz and 10 MHz settings are still global and must be applied to all channels).

Version 10.00 at Introduction

The Keysight InfiniiVision HD3-Series mixed signal oscilloscopes were introduced with version 10.00 of oscilloscope operating software.

The command set is most closely related to the InfiniiVision 3000G X-Series oscilloscopes (and the 3000/3000T X-Series, 7000A/B Series, 6000 Series, and 54620/54640 Series oscilloscopes before them). For more information, see **"Command Differences From 3000G X-Series Oscilloscopes"** on page 55.

Command Differences From 3000G X-Series Oscilloscopes

The Keysight InfiniiVision HD3-Series mixed signal oscilloscopes command set is most closely related to the InfiniiVision 3000G X-Series oscilloscopes (and the 3000/3000T X-Series, 7000A/B Series, 6000 Series, and 54620/54640 Series oscilloscopes before them).

The main differences between the version 10.00 programming command set for the InfiniiVision HD3-Series oscilloscopes and the 7.60 programming command set for the InfiniiVision 3000G X-Series oscilloscopes are related to:

- Initially, there was no Edge then Edge trigger mode in the HD3-Series oscilloscopes.
- There is no Nth Edge Burst trigger mode in the HD3-Series oscilloscopes.
- There is no TV trigger mode in the HD3-Series oscilloscopes.
- There is no USB 2.0 signal quality analysis feature or USB trigger in the HD3-Series oscilloscopes.
- There is no power measurements application feature in the HD3-Series oscilloscopes.
- Up to 20 annotations can be defined in the InfiniiVision HD3-Series oscilloscopes as compared to 10 annotations in the InfiniiVision 3000G X-Series oscilloscopes, and annotations now have a mode and can be assigned to a grid or a waveform source.
- There can be up to four graticules on the display in the InfiniiVision HD3-Series oscilloscopes as compared to one graticule in the InfiniiVision 3000G X-Series oscilloscopes.
- There are now commands for displaying the measurement results catalog, layout, and size.
- Status registers and the commands for interacting with them are now SCPI 99 compliant.
- There are two counters instead of one.
- :HARDcopy commands for setting up network printers are not included. Everything you could print in the 3000G X-Series oscilloscopes can be saved to a file on a USB storage device.
- CAN XL is supported by the CAN protocol decode.
- The ARINC 429, CXPI, FlexRay, I2S, MIL-STD-1553, Manchester, NRZ, USB, and USB PD protocol decodes are not available in the HD3-Series oscilloscopes.
- The HD3-Series oscilloscopes have a new SCPI parser with some behavior differences as compared to the SCPI parser in the 3000G X-Series oscilloscopes: see **"Commands Not Supported in Multiple Program Message Units"** on page 1713 and **"Queries Not Supported in Multiple Program Message Units"** on page 1713.

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

New Commands

Command	Description
:DIGital:ORDER:ASCending (see page 397)	Displays the digital channels that are turned on in ascending order.
:DIGital:ORDER:DESCending (see page 398)	Displays the digital channels that are turned on in descending order.
:DIGital:ORDER[:SET] (see page 399)	Displays digital channels in the order listed or returns the current order of displayed digital channels.
:DIGital<d>:SETup? (see page 402)	Returns setup information for the specified digital channel.
:DISPlay:ANNotation<n>:GRID (see page 413)	When an annotation's mode is GRID (see :DISPlay:ANNotation<n>:MODE), this command makes the association to a grid.
:DISPlay:ANNotation<n>:MODE (see page 414)	Specifies whether an annotation is associated with a grid or a waveform source.
:DISPlay:ANNotation<n>:SOURCE (see page 415)	When an annotation's mode is SOURCE (see :DISPlay:ANNotation<n>:MODE), this command makes the association to a waveform source.
:DISPlay:ANNotation<n>:XPOSITION (see page 417)	Sets the annotation's horizontal X position.
:DISPlay:ANNotation<n>:YPOSITION (see page 418)	Sets the annotation's vertical Y position.
:DISPlay:CLOCK:IGRID (see page 421)	Specifies whether the oscilloscope's clock information is displayed in the first waveform grid (ON) or in the badges bar at the top of the screen (OFF).
:DISPlay:CLOCK:IGRID:XPOSITION (see page 422)	Specifies the horizontal X position of the clock information as a percent of the grid width.
:DISPlay:CLOCK:IGRID:YPOSITION (see page 423)	Specifies the vertical Y position of the clock information as a percent of the grid height.
:DISPlay:CLOCK[:STATE] (see page 424)	Enables or disables the display of the oscilloscope's clock information.
:DISPlay:GRATICule:COUNT (see page 431)	When the graticule (grid) layout is set to CUSTOM (see :DISPlay:GRATICule:LAYOUT), this command specifies the number of waveform grids.
:DISPlay:GRATICule:ALABELS:DUAL (see page 429)	Specifies whether grid scales are shown for two waveforms.
:DISPlay:GRATICule:ALABELS:IGRID (see page 430)	Specifies whether graticule (grid) axis labels are displayed in separate scale bars (OFF) or within the grid (ON).

Command	Description
:DISPlay:GRATicule:LAYout (see page 431)	Specifies the layout of waveform display grids.
:DISPlay:GRATicule:SET (see page 431)	When the graticule (grid) layout is set to CUSTom (see :DISPlay:GRATicule:LAYout), this command assigns a waveform source to a particular grid.
:DISPlay:GRATicule:SOURce? (see page 431)	Returns a comma-separated list of the waveforms in the grid.
:DISPlay:RESults:CATalog? (see page 443)	Returns a comma-separated list of the windows in the results area.
:DISPlay:RESults:LAYout (see page 444)	Specifies the layout of windows in the results area (TAB or LIST).
:DISPlay:RESults:LAYout:LIST[:SElect] (see page 445)	Lets you select and highlight a particular window in the results area when windows are listed.
:DISPlay:RESults:LAYout:TAB:LEFT[:SElect] (see page 446)	Lets you select and highlight a particular window in the results area left pane when windows are tabbed.
:DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] (see page 447)	Lets you select and highlight a particular window in the results area right pane when windows are tabbed.
:DISPlay:RESults:SIZE (see page 448)	Specifies the size of the results area.
:FFT:BSIZE? (see page 469)	Returns the FFT's bin size value.
:FFT:DETEction:POINts (see page 472)	Specifies the maximum number of points that FFT detectors should decimate to.
:FFT:DETEction:TYPE (see page 473)	Lets you set the FFT detector decimation type for FFT (Magnitude) functions or specify that no detector is used.
:FFT:DMODE (see page 475)	Selects the FFT waveform display mode.
:FFT:RBWidth? (see page 482)	Returns the FFT's resolution bandwidth value.
:FFT:READout (see page 483)	Specifies whether the FFT resolution is displayed in the FFT spectrum plot, and if it is, whether the readout shows sample rate, bin size, or resolution bandwidth.
:FFT:SRATE? (see page 488)	Returns the FFT's sample rate value.
:FUNctio<m>:LABel (see page 557)	Sets or queries a math function's label string.
:MEASure:QUICK (see page 723)	Installs 10 common measurements on an analog input channel source.
:MEASure:THResholds:ABSolute (see page 746)	Sets the upper, middle, and lower measurement thresholds to absolute values.

Command	Description
:MEASure:THResholds:METHo d (see page 747)	Specifies whether measurement threshold values are entered as percent levels between Vbase and Vtop or as absolute (static) values.
:MEASure:THResholds:PERCen t (see page 748)	Sets the upper, middle, and lower measurement thresholds to user-defined percentages between 5% and 95% of values between Vbase and Vtop.
:MEASure:THResholds:STANda rd (see page 749)	Sets the lower, middle, and upper measurement thresholds to 10%, 50%, and 90% values between Vbase and Vtop.
:POD<n>:NIBBle<n>:THReshol d (see page 832)	Sets the logic threshold value for a nibble within a pod.
:SBUS<n>:CAN:SIGNal:XLBaud rate (see page 1013)	Sets the CAN XL baud rate.
:SBUS<n>:CAN:TRIGger:TYPE (see page 1028)	Selects the CAN trigger type.
:SBUS<n>:CAN:TYPE (see page 1029)	Selects the CAN protocol decode type.
:SBUS<n>:CAN:XLSPoint (see page 1030)	Sets the point during the CAN XL bit time where the bit level is sampled to determine whether the bit is dominant or recessive.
:STATus Commands (see page 1283)	Subsystem for controlling status registers.
:SYSTem:ERRor:ALL? (see page 1428)	Returns a comma-separated list of all errors from the queue (codes and strings).
:SYSTem:ERRor:CODE[:NEXT]? (see page 1429)	Returns the next error (code only) from the queue.
:SYSTem:ERRor:COUNt? (see page 1430)	Returns an integer number of errors in the queue.
:SYSTem:ERRor:EXTended? (see page 1431)	Returns a comma-separated list of all errors (strings, no codes) of the specified type from the queue.
:SYSTem:HELP:HEADers? (see page 1434)	Returns a binary block that contains a newline-separated list of supported SCPI commands and queries.
:SYSTem:HID? (see page 1434)	Returns the oscilloscope's Host ID string (the same as :SYSTem:DIDentifier?).
:SYSTem:POWercycle (see page 1442)	Powers-down the oscilloscope and restarts it. This is the same as pressing the front panel power button to power-down the oscilloscope and then pressing the power button again to start the oscilloscope.
:SYSTem:SHUTdown (see page 1459)	Powers-down the oscilloscope. This is the same as pressing the front panel power button to power-down the oscilloscope.
:SYSTem:TIME:DSAVing (see page 1461)	Enables or disables the "automatically adjust clock for Daylight Savings Time" setting.

Command	Description
:SYSTem:TIME:NTPProtocol (see page 1462)	Enables or disables automatically setting the oscilloscope's clock using Network Time Protocol.
:SYSTem:TIME:ZONE (see page 1463)	Sets the system time zone based on the specified offset from UTC.
:WGEN<w>:DCMode (see page 1605)	When the DC waveform type is selected, this command lets you choose between PRECise output levels using a smaller range (± 1.00 V) or a WIDerange of voltages (± 8.00 V).

Changed Commands

Command	Differences From InfiniiVision 3000G X-Series Oscilloscopes
:ACQuire:TYPE (see page 311)	There is no HREsolution acquisition type. Instead, higher vertical resolutions are achieved by selecting lower bandwidth limits in the :CHANnel<n>:BANDwidth (see page 349) command.
:CALibrate:LABel (see page 326)	The :CALibrate:LABel? query now returns the string within quotes.
:CALibrate:PROTected (see page 328)	There is now a command form that disables or enables the Cal Protect control which allows user calibration to be run or prevents user calibration from being run.
:DISPlay:DATA? (see page 425)	The option for selecting a color or grayscale palette is no longer available. Also, the BMP8bit option is no longer available.
:MTESt:TITLe? (see page 827)	The query now returns the string within quotes.
*RCL (see page 246)	The *RCL command is similar to the :RECall:SETup[:START] (see page 950) command and will recall setup files from internal locations under the "/User Files/" folder as well as from external USB storage device locations under the "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" folders depending on the number of storage devices connected.
:RECall:ARBitrary[:START] (see page 944)	When recalling from internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
:RECall:MASK[:START] (see page 948)	When recalling from internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
:RECall:SETup[:START] (see page 950)	When recalling from internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
*SAV (see page 250)	The *SAV command is similar to the :SAVE:SETup[:START] (see page 974) command and will save setup files to internal locations under the "/User Files/" folder as well as to external USB storage device locations under the "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" folders depending on the number of storage devices connected.

Command	Differences From InfiniiVision 3000G X-Series Oscilloscopes
:SAVE:ARbitrary[:START] (see page 957)	When saving to internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
:SAVE:MASK[:START] (see page 963)	When saving to internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
:SAVE:SETup[:START] (see page 974)	When saving to internal locations, quoted ASCII file name strings beginning with "/User Files/" are used instead of internal location numbers.
:SBUS<n>:CAN:TRIGger (see page 1015)	Separate FDData and FDMSignal options are no longer necessary with the addition of the :SBUS<n>:CAN:TRIGger:TYPE command.
:SBUS<n>:IIC:TRIGger[:TYPE] (see page 1039)	The "READEprom" option has changed to "REPRom".
:SEARch:SERial:IIC:MODE (see page 1221)	The "READEprom" option has changed to "REPRom".
:SERial? (see page 276)	The query now returns the string within quotes.
:SYSTem:ERRor[:NEXT]? (see page 1432)	The query is the same except the optional [:NEXT] can be included.
:SYSTem:PERSonal[:MANufacturer] (see page 1440)	The :SYSTem:PERSonal[:MANufacturer]? query now returns the string within quotes.

Discontinued Commands

Discontinued commands are commands that were available in InfiniiVision 3000G X-Series oscilloscopes, but are not supported by the InfiniiVision HD3-Series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

Discontinued Command	Current Command Equivalent	Comments
:CHANnel2:SKEW	:CHANnel<n>:PROBe:SKEW (see page 369)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:ACTivity	:ACTivity (see page 262)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:LABel	:CHANnel<n>:LABel (see page 354) or :DIGital<n>:LABel (see page 401)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:THReshold	:POD<n>:THReshold (see page 835) or :DIGital<d>:THReshold (see page 404)	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:CHANnel<n>:INPut	:CHANnel<n>:IMPedance (see page 352)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel<n>:PMODE	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:DIGItal<d>:POSition	:DIGItal:ORDER[:SET] (see page 399)	The notion of digital channel position locations is not present in the HD3-Series oscilloscopes. You can, however, order digital channels using the :DIGItal:ORDER[:SET] (see page 399) command.
:DISPlay:ANNotation<n>:X1Pos ition	:DISPlay:ANNotation<n>:XPOSi tion (see page 417)	Discontinued command positioned by pixel position, current command positions by percent of grid width.
:DISPlay:ANNotation<n>:Y1Pos ition	:DISPlay:ANNotation<n>:YPOSi tion (see page 418)	Discontinued command positioned by pixel position, current command positions by percent of grid height.
:DISPlay:CONNEct	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:DISPlay:GRATicule:TYPE	none	This command was for the TV trigger mode which is not available in the HD3-Series oscilloscopes.
:DISPlay:MENU	none	On earlier InfiniiVision oscilloscopes, this command would display a particular softkey menu. The HD3-Series oscilloscopes do not have softkey menus.
:DISPlay:ORDer	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:DISPlay:SIDebar	none	On earlier InfiniiVision oscilloscopes, this command would display a particular sidebar dialog box. The HD3-Series oscilloscopes do not have sidebar dialog boxes.

Discontinued Command	Current Command Equivalent	Comments
:DVM:FREQuency?	:COUNter<c>:CURRent? (see page 387)	The :DVM:FREQuency? query has been replaced by the new Chapter 14 , “:COUNter<c> Commands,” starting on page 385.
:ERASe	:DISplay:CLEar (see page 420)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:EXTernal:PMODE	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:FFT:DMODE	none	FFT display mode options are not available in the HD3-Series oscilloscopes. However, You can enable an FFT Max Function in the user interface.
:FUNction:GOFT:OPERation	:FUNction1:OPERation (see page 561)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:GOFT:SOURce1	:FUNction1:SOURce1 (see page 569)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:GOFT:SOURce2	:FUNction1:SOURce2 (see page 571)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:SOURce	:FUNction:SOURce1 (see page 569)	Obsolete command has ADD, SUBTract, and MULTiPLY parameters; current command has GOFT parameter. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:VIEW	:FUNction:DISPlay (see page 533)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:DESTination	:RECall:FILeName (see page 946) :SAVE:FILeName (see page 946)	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:HARDcopy:FILENAME	:RECall:FILENAME (see page 946) :SAVE:FILENAME (see page 946)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:GRAYscale	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:IGColors	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:PDRiver	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HWEenable	:STATus:OPERation:HARDware:ENABLE (see page 1325)	Status registers now use SCPI standard commands and queries for controlling and reading the registers.
:HWERegister:CONDition?	:STATus:OPERation:HARDware:CONDition? (see page 1324)	
:HWERegister[:EVENT]?	:STATus:OPERation:HARDware[:EVENT]? (see page 1328)	
:MEASure:ALL	:MEASure:QUICK (see page 723)	There is no "Snapshot All" feature in the HD3-Series oscilloscopes, but there is a "Quick Measure" that will install 10 common measurements on an analog input channel source.
:MEASure:DEFine	:MEASure:DELay:DEFine (see page 685) :MEASure:THResholds:ABSolute (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:DEFine functionality is provided by the current command equivalents.

Discontinued Command	Current Command Equivalent	Comments
:MEASure:LOWer	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TDELta	:MARKer:XDELta (see page 642)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:THResholds	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TMAX	:MEASure:XMAX (see page 766)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TMIN	:MEASure:XMIN (see page 767)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TSTArt	:MARKer:X1Position (see page 637)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TSTOp	:MARKer:X2Position (see page 640)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TVOLt	:MEASure:TVALue (see page 754)	TVALue measures additional values such as db, Vs, etc. This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:MEASure:UPPer	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VDELta	:MARKer:YDELta (see page 649)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VSTArt	:MARKer:Y1Position (see page 646)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VSTOp	:MARKer:Y2Position (see page 648)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VTIME	:MEASure:YATX (see page 768)	With :MEASure:YATX, there is a command for installing the measurement on the oscilloscope's display. :MEASure:VTIME had no command, only a query.
:MTEenable	:STATus:OPERation:MTEST:ENA Ble (see page 1351)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:MTERegister[:EVENT]?	:STATus:OPERation:MTEST[:EVE Nt]? (see page 1354)	
:MTEST:AMASK:{SAVE STORE}	:SAVE:MASK[:START] (see page 963)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:AVERage	:ACQuire:TYPE AVERage (see page 311)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:AVERage:COUNt	:ACQuire:COUNt (see page 288)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:LOAD	:RECall:MASK[:START] (see page 948)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:RUMode	:MTEST:RMODE (see page 813)	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:MTESt:RUMode:SOFailure	:MTESt:RMODE:FACTion:STOP (see page 817)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTESt:{START STOP}	:RUN (see page 275) or :STOP (see page 279)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTESt:TRIGger:SOURce	:TRIGger Commands (see page 1479)	There are various commands for setting the source with different types of triggers. This was an obsolete command in the 3000G X-Series oscilloscopes.
:OPEE	:STATus:OPERation:ENABLE (see page 1299)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:OPERRegister:CONDition?	:STATus:OPERation:CONDition? (see page 1298)	
:OPERRegister[:EVENT]?	:STATus:OPERation[:EVENT]? (see page 1302)	
:OVLenable	:STATus:OPERation:OVERload:ENABLE (see page 1364)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:OVLRegister?	:STATus:OPERation:OVERload[:EVENT]? (see page 1367)	
:PRINt	:DISPlay:DATA? (see page 425)	Printers are not supported in the HD3-Series oscilloscopes. Everything you could print in the 3000G X-Series oscilloscopes can be saved to a file on a USB storage device. The :PRINt? query was a obsolete in the 3000G X-Series oscilloscopes.
:SAVE:IMAGe:AREA	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SAVE:IMAGe:PALette	none	Color images are always saved in the HD3-Series oscilloscopes.
:SBUS<n>:CAN:FDSTandard	none	This is no longer an option. The ISO standard is used when decoding or triggering on FD frames.

Discontinued Command	Current Command Equivalent	Comments
:SBUS<n>:LIN:SIGNAL:DEFinition	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SYSTEM:MENU	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SYSTEM:PRECision	:SYSTEM:PRECision:LENGTH (see page 1443)	Precision analysis is no longer a feature to be enabled. Instead, you simply choose the desired analysis record length.
:TRIGGER:THReshold	:POD<n>:THReshold (see page 835) or :DIGital<d>:THReshold (see page 404)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:TRIGGER:ZONE:STATE	none	There is no longer an overall zone trigger feature on/off state.
:WAVEform:VIEW	none	In the HD3-Series oscilloscopes, all captured data is used.

2 Setting Up

Step 1. Install Keysight IO Libraries Suite software / 70

Step 2. Connect and set up the oscilloscope / 71

Step 3. Verify the oscilloscope connection / 73

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 HD3-Series oscilloscope has two different interfaces you can use for programming:

- USB (device port).
- LAN. To configure the LAN interface, see **"Using the LAN Interface"** on page 71.

These interfaces are always active.

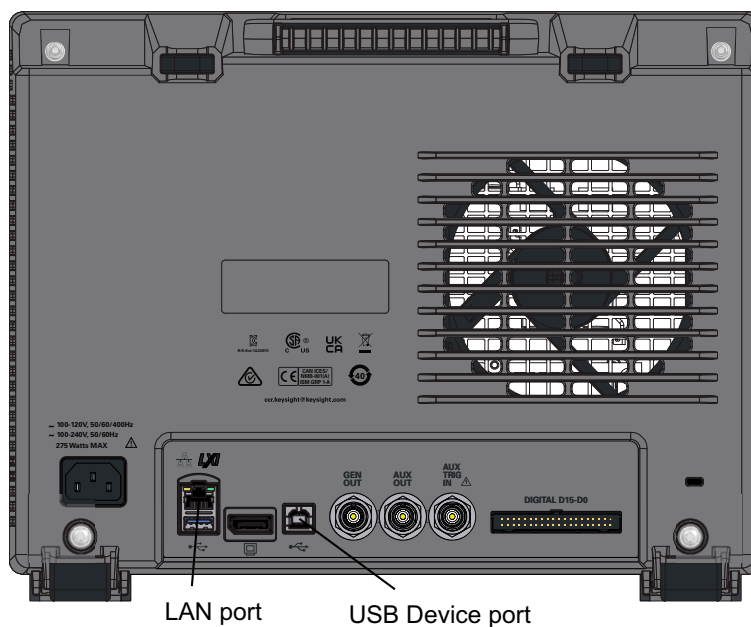


Figure 1 Control Connectors on Rear Panel

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.

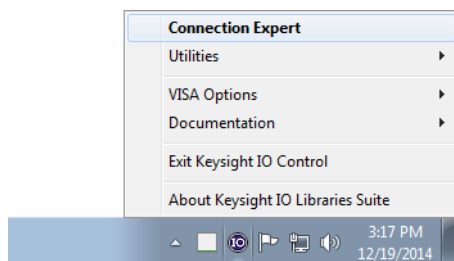
Find out if automatic configuration via DHCP or AutoIP can be used. Also, find out whether your network supports Dynamic DNS or Multicast DNS.

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

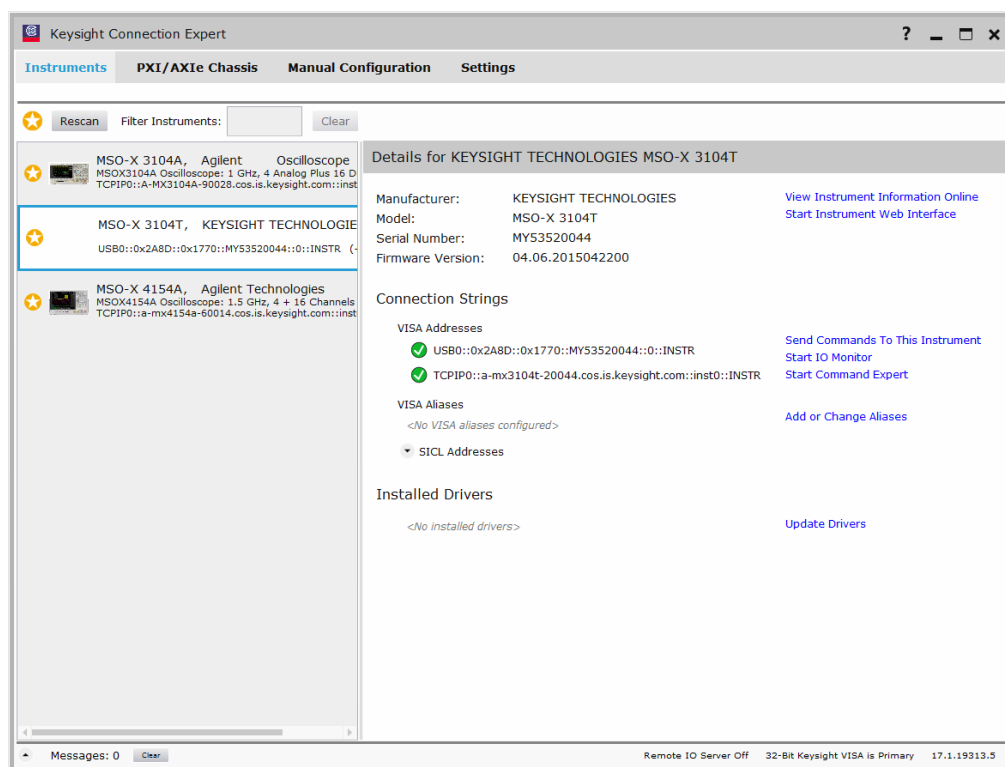
- 3 Connect the oscilloscope to the local area network (LAN) by inserting LAN cable into the "LAN" port.
- 4 Configure the oscilloscope's LAN interface:
 - a From the menu, choose **Utilities > I/O....**
 - b In the I/O dialog box's **LAN Setup** tab, select **LAN Settings....**
 - c In the LAN Settings dialog box:
 - If automatic configuration via DHCP or AutoIP can be used, select **Automatic**.
 - If your network supports Multicast DNS, select **Multicast DNS**.
 - If automatic configuration is not supported, clear **Automatic**. Then, enter the appropriate values in the **IP Address**, **Subnet Mask**, **Gateway IP**, and **DNS IP** fields.
 - d Enter the oscilloscope's **Host Name**.
 - e Select **Apply**.

Step 3. Verify the oscilloscope connection

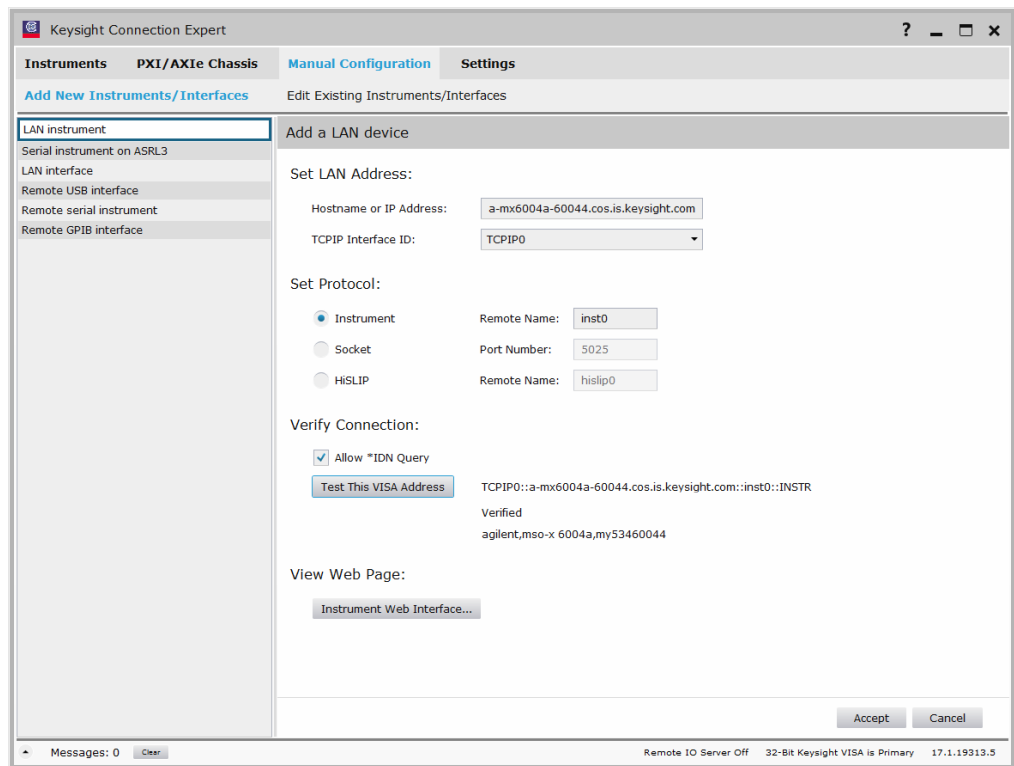
- 1 On the controller PC, click on the Keysight IO Control icon in the taskbar and choose **Connection Expert** from the popup menu.



- 2 In the Keysight Connection Expert application, instruments connected to the controller's USB interface 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.

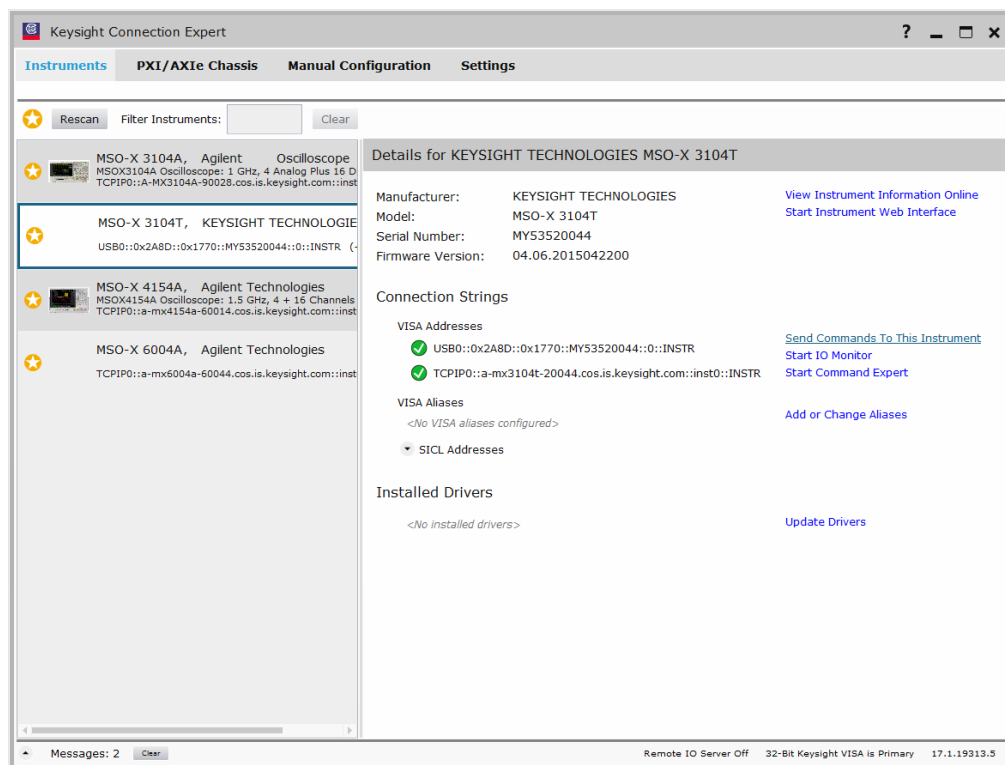


For example, to add a device:

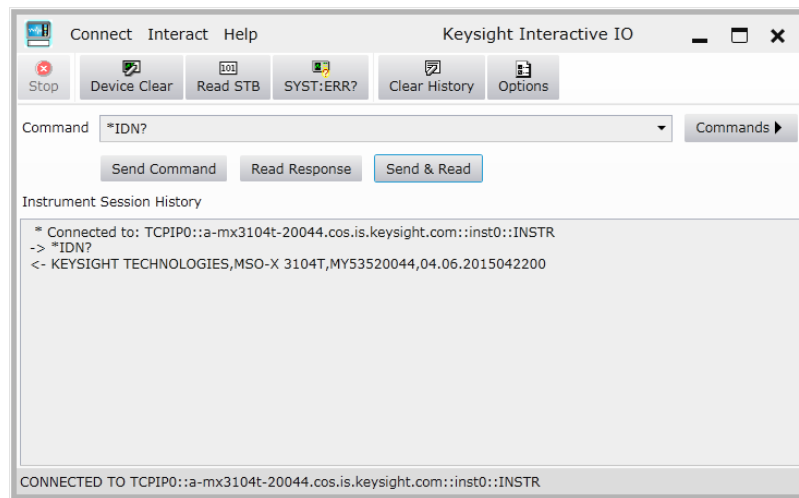
- a Select **LAN instrument** in the list on the left.
- b Enter the oscilloscope's **Hostname** or **IP address**.
- c Select the protocol.
- d Select **Instrument** under Set Protocol.
- e Click **Test This VISA Address** to verify the connection.
- f If the connection test is successful, click **Accept**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



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



- 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 / 78

Programming the Oscilloscope / 80

Other Ways of Sending Commands / 89

This chapter gives you an overview of programming the HD3-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.

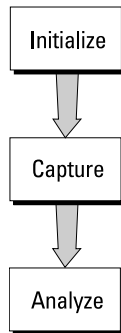
NOTE

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 80
- "Opening the Oscilloscope Connection via the IO Library" on page 81
- "Using :AUToscale to Automate Oscilloscope Setup" on page 82
- "Using Other Oscilloscope Setup Commands" on page 82
- "Capturing Data with the :DIGitize Command" on page 83
- "Reading Query Responses from the Oscilloscope" on page 85
- "Reading Query Results into String Variables" on page 85
- "Reading Query Results into Numeric Variables" on page 86
- "Reading Definite-Length Block Query Response Data" on page 86
- "Sending Multiple Queries and Reading Results" on page 87
- "Checking Instrument Status" on page 88

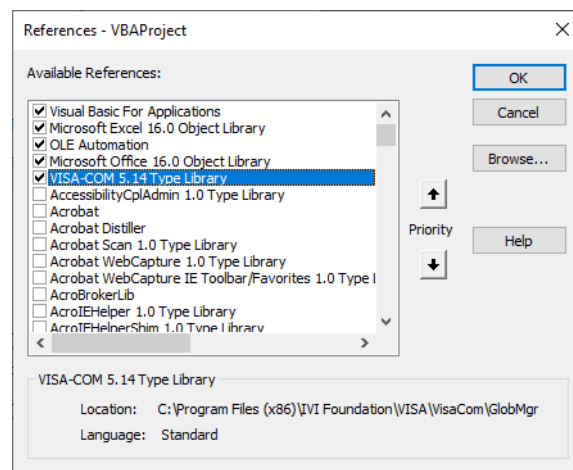
Referencing the IO Library

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 5.14 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 5.14 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 1709.

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 20 seconds
.
myScope.IO.Clear
myScope.IO.Timeout = 20000

```

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 6](#), “Common (*) Commands,” starting on page 231.

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 μ 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 = 20000    ' Set interface timeout to 20 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 ":CHANnel:PROBe 10"        ' Probe attenuation
                                                ' to 10:1.
myScope.WriteString ":CHANnel:RANGe 1.6"        ' Vertical range
                                                ' 1.6 V full scale.
myScope.WriteString ":CHANnel:OFFSet -0.4"      ' Offset to -0.4.
myScope.WriteString ":CHANnel: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:COMPLet 100"
myScope.WriteString ":ACQuire:COUNt 8"
myScope.WriteString ":DIGitize CHANnel1"
myScope.WriteString ":WAVeform:SOURce CHANnel1"
myScope.WriteString ":WAVeform:FORMat WORD"
myScope.WriteString ":WAVeform:BYTeorder LSBF"
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 41](#), ":WAVeform Commands," starting on page 1553.

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:

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

Number of Digits That Follow: 8
Actual Data: <1000 bytes of data>
Number of Bytes to be Transmitted: 1000

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(ASCIIType_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"
```

See Also • **"Multiple Program Message Units"** on page 1713

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 46**, "Status Reporting," starting on page 1669 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 Instrument IO:

- **"Telnet Sockets"** on page 89
- **"Sending SCPI Commands Using Browser Instrument IO"** on page 89

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 Instrument IO

To send SCPI commands using the browser Instrument IO feature, establish a connection to the oscilloscope via LAN as described in the *InfiniiVision HD3-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 **Control Instrument** tab, then select **Use Instrument IO**.

4 Sequential (Blocking) vs. Overlapped Commands

IEEE 488.2 makes the distinction between sequential and overlapped commands (and queries):

- *Sequential commands* also known as *blocking commands*, finish their task before the execution of the next command starts.

These oscilloscope commands and queries are sequential (blocking):

- :AUToscale
- :DIGitize
- :WMEMemory<r>:SAVE <source>
- :WAVEform:DATA?
- *Overlapped commands* run concurrently. Commands following an overlapped command may be started before the overlapped command is completed.

Some oscilloscope commands are overlapped. For example, the oscilloscope's save and recall commands are overlapped as well as some commands that perform analysis.

With sequential (blocking) commands and queries, the oscilloscope is expected to stop processing inputs, including additional remote commands and queries as well as front panel knobs, until completed.

Pausing Execution Between Overlapped Commands

With overlapped commands, you can use the *OPC? query to prevent any more commands from being executed until the overlapped command is complete. This may be necessary when a command that follows an overlapped command interferes with the overlapped command's processing or analysis. For example:

```
:RECall:SETup "setup.scp";*OPC?;:RECall:ARbitrary "arb_wfm.csv"
```

You can also use the *ESR? query to look at the OPC bit (bit 0) in the Standard Event Status Register to determine when an operation is complete.

Pausing Programs Until Sequential (Blocking) Commands are Complete

Sequential (blocking) commands do not prevent additional commands from being sent to the queue or cause the remote program to wait. For example, if your program does something like:

```
myScope.WriteString ":DIGitize"
Debug.Print "Signal acquired."
```

The "Signal acquired" message will be written immediately after the :DIGitize is sent, not after the acquisition and processing is complete.

To pause the program until a sequential (blocking) command is complete, you must wait for a query result after the sequential (blocking) command. For example, in this case:

```
Public strQueryResult As String
myScope.WriteString ":DIGitize;*OPC?"
strQueryResult = myScope.ReadString
Debug.Print "Signal acquired."
```

The "Signal acquired" message will be written after the acquisition and processing is complete. The *OPC? query is appended to :DIGitize with a semi-colon (;), which essentially ties it to the same thread in the parser. It is immediately dealt with once :DIGitize finishes and gives a "1" back to the program (whether the program uses it or not), allowing the program to move on.

When using a query to wait until a sequential (blocking) command is complete, it is possible for the sequential (blocking) command execution to take longer than the I/O timeout, in which case, there will be a timeout error while waiting for the query results. You can increase the I/O timeout or have your program poll the Status Byte Register using the IO libraries' unblocked ReadSTB function to wait for execution completion.

Using Device Clear to Abort a Sequential (Blocking) Command

When sequential (blocking) commands take too long or fail to complete for some reason, you can send a Device Clear over the bus to clear the input buffer and output queue, reset the parser, and clear any pending commands.

- See Also
- **"*OPC (Operation Complete)"** on page 243
 - **"*ESR? (Standard Event Status Register)"** on page 239
 - **Chapter 47**, "Synchronizing Acquisitions," starting on page 1685

5 Commands Quick Reference

Command Summary / 94

Syntax Elements / 227

Command Summary

- Common (*) Commands Summary (see [page 96](#))
- Root (:) Commands Summary (see [page 100](#))
- :ACQuire Commands Summary (see [page 102](#))
- :BUS<n> Commands Summary (see [page 104](#))
- :CALibrate Commands Summary (see [page 105](#))
- :CGRade Commands Summary (see [page 105](#))
- :CHANnel<n> Commands Summary (see [page 107](#))
- :CLOCK Commands Summary (see [page 110](#))
- :COUNter Commands Summary (see [page 111](#))
- :DIGital<n> Commands Summary (see [page 111](#))
- :DISPlay Commands Summary (see [page 112](#))
- :DVM Commands Summary (see [page 116](#))
- :EXTernal Trigger Commands Summary (see [page 117](#))
- :FFT Commands Summary (see [page 117](#))
- :FRANalysis Commands Summary (see [page 119](#))
- :FUNction Commands Summary (see [page 120](#))
- :HCOPy Commands Summary (see [page 126](#))
- :HISTogram Commands Summary (see [page 127](#))
- :JITter Commands Summary (see [page 128](#))
- :LISTer Commands Summary (see [page 129](#))
- :LTESr Commands Summary (see [page 129](#))
- :MARKer Commands Summary (see [page 130](#))
- :MEASure Commands Summary (see [page 132](#))
- :MEASure Power Commands Summary (see [page 151](#))
- :MTEST Commands Summary (see [page 155](#))
- :POD<n> Commands Summary (see [page 158](#))
- :POWer Commands Summary (see [page 158](#))
- :RECall Commands Summary (see [page 166](#))
- :SAVE Commands Summary (see [page 168](#))
- General :SBUS<n> Commands Summary (see [page 170](#))
- :SBUS<n>:A429 Commands Summary (see [page 171](#))
- :SBUS<n>:CAN Commands Summary (see [page 172](#))
- :SBUS<n>:IIC Commands Summary (see [page 174](#))

- :SBUS<n>:LIN Commands Summary (see [page 175](#))
- :SBUS<n>:M1553 Commands Summary (see [page 177](#))
- :SBUS<n>:SENT Commands Summary (see [page 177](#))
- :SBUS<n>:SPI Commands Summary (see [page 179](#))
- :SBUS<n>:UART Commands Summary (see [page 181](#))
- :SBUS<n>:USB Commands Summary (see [page 183](#))
- :SBUS<n>:USBPd Commands Summary (see [page 185](#))
- General :SEARch Commands Summary (see [page 186](#))
- :SEARch:EDGE Commands Summary (see [page 187](#))
- :SEARch:GLITCh Commands Summary (see [page 187](#))
- :SEARch:PEAK Commands Summary (see [page 188](#))
- :SEARch:RUNT Commands Summary (see [page 188](#))
- :SEARch:TRANSition Commands Summary (see [page 189](#))
- :SEARch:SERial:A429 Commands Summary (see [page 189](#))
- :SEARch:SERial:CAN Commands Summary (see [page 190](#))
- :SEARch:SERial:IIC Commands Summary (see [page 190](#))
- :SEARch:SERial:LIN Commands Summary (see [page 191](#))
- :SEARch:SERial:M1553 Commands Summary (see [page 192](#))
- :SEARch:SERial:SENT Commands Summary (see [page 192](#))
- :SEARch:SERial:SPI Commands Summary (see [page 193](#))
- :SEARch:SERial:UART Commands Summary (see [page 193](#))
- :SEARch:SERial:USB Commands Summary (see [page 194](#))
- :SEARch:SERial:USBPd Commands Summary (see [page 195](#))
- General :STATus Commands Summary (see [page 196](#))
- :STATus:OPERation Commands Summary (see [page 196](#))
- :STATus:OPERation:ARM Commands Summary (see [page 197](#))
- :STATus:OPERation:HARDware Commands Summary (see [page 198](#))
- :STATus:OPERation:LOCal Commands Summary (see [page 199](#))
- :STATus:OPERation::MTESt Commands Summary (see [page 200](#))
- :STATus:OPERation:OVERload Commands Summary (see [page 201](#))
- :STATus:OPERation:OVERload:PFault Commands Summary (see [page 202](#))
- :STATus:OPERation::POWEr Commands Summary (see [page 204](#))
- :STATus:TRIGger Commands Summary (see [page 205](#))
- :STATus:USER Commands Summary (see [page 206](#))
- :SYSTem Commands Summary (see [page 207](#))

- :TImEbase Commands Summary (see [page 210](#))
- General :TRIGger Commands Summary (see [page 211](#))
- :TRIGger:DELAy Commands Summary (see [page 213](#))
- :TRIGger[:EDGE] Commands Summary (see [page 214](#))
- :TRIGger:GLITCh Commands Summary (see [page 215](#))
- :TRIGger:OR Commands Summary (see [page 216](#))
- :TRIGger:PATtern Commands Summary (see [page 216](#))
- :TRIGger:RUNT Commands Summary (see [page 217](#))
- :TRIGger:SHOLd Commands Summary (see [page 217](#))
- :TRIGger:TRANSition Commands Summary (see [page 218](#))
- :TRIGger:ZONE Commands Summary (see [page 219](#))
- :WAVEform Commands Summary (see [page 219](#))
- :WGEN Commands Summary (see [page 222](#))
- :WMEMory<r> Commands Summary (see [page 225](#))

Table 2 Common (*) Commands Summary

Command	Query	Options and Query Returns
*CLS (see page 236)	n/a	n/a
*ESE <mask> (see page 237)	*ESE? (see page 238)	<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 page 239)	<status> ::= 0 to 255; an integer in NR1 format

Table 2 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	*IDN? (see page 239)	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 page 242)	<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format
*OPC (see page 243)	*OPC? (see page 243)	ASCII "1" is placed in the output queue when all pending device operations have completed.
n/a	*OPT? (see page 244)	<return_value> ::= 0,0,<license info> <license info> ::= <All field>, <reserved>, <Enhanced Security>, <Waveform Generator>, <Automotive Serial>, <Aerospace Serial>, <Embedded Serial>, <USB Serial>, <Power Measurements>, <Jitter Analysis>, <reserved>, <reserved>, <reserved>, <Memory 100M>, <Memory 50M>, <Memory 20M>, <Bandwidth 1GHz>, <Bandwidth 500MHz>, <Bandwidth 350MHz>, <Bandwidth 200MHz>, <Demo Partner>, <Demo Keysight>

Table 2 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	*OPT? (see page 244) (cont'd)	<All field> ::= {0 All(d)} <reserved> ::= 0 <Enhanced Security> ::= {0 HD3SECURE} <Waveform Generator> ::= {0 WAVEGEN} <Automotive Serial> ::= {0 HD300AUTA} <Aerospace Serial> ::= {0 HD300AERA} <Embedded Serial> ::= {0 HD300EMBA} <USB Serial> ::= {0 HD300USBA} <Power Measurements> ::= {0 HD300PWRA} <Jitter Analysis> ::= {0 HD300QJTA} <Memory 100M> ::= {0 MEMUP100} <Memory 50M> ::= {0 MEMUP50} <Memory 20M> ::= {0 MEMUP20} <Bandwidth 1GHz> ::= {0 BW1000} <Bandwidth 500MHz> ::= {0 BW500} <Bandwidth 350MHz> ::= {0 BW350} <Bandwidth 200MHz> ::= {0 BW200} <Demo Partner> ::= {0 DIS} <Demo Keysight> ::= {0 D24}
*RCL {<internal_fname> <external_fname>} (see page 246)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
*RST (see page 247)	n/a	See *RST (Reset) (see page 247)

Table 2 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
*SAV {<internal_fname> <external_fname>} (see page 250)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
*SRE <mask> (see page 251)	*SRE? (see page 252)	<mask> ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. <mask> ::= following values: 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 253)	<value> ::= 0 to 255; an integer in NR1 format, as shown in the following: 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 256)	n/a	n/a
n/a	*TST? (see page 257)	<result> ::= 0 or non-zero value; an integer in NR1 format
*WAI (see page 258)	n/a	n/a

Table 3 Root (:) Commands Summary

Command	Query	Options and Query Returns
:ACTivity (see page 262)	:ACTivity? (see page 262)	<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 263)	{0 1}; an integer in NR1 format
:AUToscale [<source>[,...,<source>]] (see page 265)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD1 POD2} <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 267)	:AUToscale:AMODE? (see page 267)	<value> ::= {NORMal CURRent}}
:AUToscale:CHANnels <value> (see page 268)	:AUToscale:CHANnels? (see page 268)	<value> ::= {ALL DISPlayed}}
:AUToscale:FDEBug { {0 OFF} {1 ON} } (see page 269)	:AUToscale:FDEBug? (see page 269)	{0 1}
:BLANK [<source>] (see page 270)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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

Table 3 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGitize [<source>[,...,<source>]] (see page 271)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} SBUS{1 2}}} <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 <m> ::= 1 to (# math functions) in NR1 format
n/a	:MODEl? [FRAME] (see page 276)	<return value> ::= string containing the oscilloscope's model number
n/a	:RSTate? (see page 274)	n/a
:RUN (see page 275)	n/a	n/a
n/a	:SERial? (see page 276)	<return value> ::= quoted string containing serial number
:SINGle (see page 277)	n/a	n/a
n/a	:STATus? <display> (see page 278)	{0 1} <display> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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 page 279)	n/a	n/a

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

Command	Query	Options and Query Returns
n/a	:TER? (see page 280)	{0 1}
:VIEW <source> (see page 281)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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

Table 4 :ACquire Commands Summary

Command	Query	Options and Query Returns
n/a	:ACquire:AAlias? (see page 286)	{1 0}
:ACquire:COMplete <complete> (see page 287)	:ACquire:COMplete? (see page 287)	<complete> ::= 100; an integer in NR1 format
:ACquire:COUNt <count> (see page 288)	:ACquire:COUNt? (see page 288)	<count> ::= an integer from 2 to 65536 in NR1 format
:ACquire:DAAlias <mode> (see page 289)	:ACquire:DAAlias? (see page 289)	<mode> ::= {DISable AUTO}
:ACquire:DIGitizer {{0 OFF} {1 ON}} (see page 290)	:ACquire:DIGitizer? (see page 290)	{0 1}
:ACquire:MANual {{0 OFF} {1 ON}} (see page 292)	:ACquire:MANual? (see page 292)	{0 1}
n/a	:ACquire:MANual:POINTs? (see page 294)	<gui_setting> ::= Max Memory Depth setting in NR1 format
n/a	:ACquire:MANual:SRATe? (see page 295)	<gui_setting> ::= Max Sample Rate setting in NR3 format
:ACquire:MODE <mode> (see page 296)	:ACquire:MODE? (see page 296)	<mode> ::= {RTIME ETIME SEGmented}

Table 4 :ACQUIRE Commands Summary (continued)

Command	Query	Options and Query Returns
:ACQUIRE:POINTS[:ANALog] <points> (see page 297)	:ACQUIRE:POINTS[:ANALog]? (see page 297)	<points> ::= {AUTO <points_value>} <points_value> ::= desired analog memory depth in integer NR1 format
:ACQUIRE:POINTS[:ANALog]:AUTO {{0 OFF} {1 ON}} (see page 298)	:ACQUIRE:POINTS[:ANALog]:AUTO? (see page 299)	{0 1}
n/a	:ACQUIRE:POINTS:MAX? (see page 301)	<max_points> ::= an integer in NR1 format
:ACQUIRE:SEGMENTED:ANALyze (see page 302)	n/a	n/a
:ACQUIRE:SEGMENTED:COUNT <count> (see page 303)	:ACQUIRE:SEGMENTED:COUNT? (see page 303)	<count> ::= an integer from 2 to 1000 in NR1 format
:ACQUIRE:SEGMENTED:INDEX <index> (see page 304)	:ACQUIRE:SEGMENTED:INDEX? (see page 304)	<index> ::= an integer from 1 to 1000 in NR1 format
:ACQUIRE:SRATE[:ANALog] <rate> (see page 307)	:ACQUIRE:SRATE[:ANALog]? (see page 307)	<rate> ::= {AUTO <sample_rate>} <sample_rate> ::= desired analog sample rate in NR3 format
:ACQUIRE:SRATE[:ANALog]:AUTO {{0 OFF} {1 ON}} (see page 308)	:ACQUIRE:SRATE[:ANALog]:AUTO? (see page 309)	{0 1}
:ACQUIRE:TYPE <type> (see page 311)	:ACQUIRE:TYPE? (see page 311)	<type> ::= {NORMAL AVERAGE PEAK}

Table 5 :BUS<n> Commands Summary

Command	Query	Options and Query Returns
:BUS<n>:BIT<m> {{0 OFF} {1 ON}} (see page 315)	:BUS<n>:BIT<m>? (see page 315)	{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 316)	:BUS<n>:BITS? (see page 316)	<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 318)	n/a	<n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:DISPlay {{0 OFF} {1 ON}} (see page 319)	:BUS<n>:DISPlay? (see page 319)	{0 1} <n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:LABel <string> (see page 320)	:BUS<n>:LABel? (see page 320)	<string> ::= quoted ASCII string up to 32 characters <n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:MASK <mask> (see page 321)	:BUS<n>:MASK? (see page 321)	<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 page 325)	<return value> ::= <year>,<month>,<day>; all in NR1 format
:CALibrate:LABel <string> (see page 326)	:CALibrate:LABel? (see page 326)	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:OUTPut <signal> (see page 327)	:CALibrate:OUTPut? (see page 327)	<signal> ::= {TRIGgers MASK WAVEgen WGEN1 TSource} Note: WAVE and WGEN1 are equivalent.
:CALibrate:PROTected {{0 OFF} {1 ON}} (see page 328)	:CALibrate:PROTected? (see page 328)	{"PROTected" "UNPRotected"}
:CALibrate:START (see page 329)	n/a	n/a
n/a	:CALibrate:STATus? (see page 330)	<return value> ::= <status_code>,<status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:TEMPeratur e? (see page 331)	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see page 332)	<return value> ::= <hours>,<minutes>,<seconds>; all in NR1 format

Table 7 :CGRade Commands Summary

Command	Query	Options and Query Returns
:CGRade:ANALog:ALL {{0 OFF} {1 ON}} (see page 335)	:CGRade:ANALog:ALL? (see page 335)	{0 1}
:CGRade:INVert {{0 OFF} {1 ON}} (see page 336)	:CGRade:INVert? (see page 336)	{0 1}

Table 7 :CGRade Commands Summary (continued)

Command	Query	Options and Query Returns
:CGRade:KEY:SOURce <source> (see page 337)	:CGRade:KEY:SOURce? (see page 337)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1-4 in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format
n/a	:CGRade:LEVelS? (see page 338)	<levels> ::= comma-delimited list of color range levels
:CGRade:LEVelS:DISPla y {{0 OFF} {1 ON}} (see page 339)	:CGRade:LEVelS:DISPla y? (see page 339)	{0 1}
:CGRade:MATH:SOURce:S TATe {{0 OFF} {1 ON}} (see page 340)	:CGRade:MATH:SOURce:S TATe? (see page 340)	{0 1}
:CGRade:RESet (see page 341)	n/a	n/a
:CGRade:SCHeMe <theme> (see page 342)	:CGRade:SCHeMe? (see page 342)	<theme> ::= {TEMPerature INTensity VIRidis HOT ABYSS SKY INFerno}
:CGRade:SOURce <source> (see page 343)	:CGRade:SOURce? (see page 343)	<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
:CGRade:STATe {{0 OFF} {1 ON}} (see page 344)	:CGRade:STATe? (see page 344)	{0 1}

Table 8 :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BANDwidth <channel_bw_limit> (see page 349)	:CHANnel<n>:BANDwidth? (see page 349)	{1000000000 350000000 200000000 100000000 50000000 20000000 10000000 5000000} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:COUpling <coupling> (see page 350)	:CHANnel<n>:COUpling? (see page 350)	<coupling> ::= {AC DC} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:DISPlay {{0 OFF} {1 ON}} (see page 351)	:CHANnel<n>:DISPlay? (see page 351)	{0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:IMPedance <impedance> (see page 352)	:CHANnel<n>:IMPedance? (see page 352)	<impedance> ::= {ONEMeg FIFTy} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:INVert {{0 OFF} {1 ON}} (see page 353)	:CHANnel<n>:INVert? (see page 353)	{0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:LABel <string> (see page 354)	:CHANnel<n>:LABel? (see page 354)	<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 page 355)	:CHANnel<n>:OFFSet? (see page 355)	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V mV} <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe <attenuation> (see page 356)	:CHANnel<n>:PROBe? (see page 356)	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe:BTN <setting> (see page 357)	:CHANnel<n>:PROBe:BTN? (see page 357)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {HEADlight INFinimode RSTop SINGle CDISplay AUToscale FTRigger QACTion NACTion NONE}
:CHANnel<n>:PROBe:CAL ibration (see page 358)	n/a	n/a

Table 8 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:CHANnel<n>:PROBe:CALibration:VALidity? (see page 359)	<n> ::= 1 to (# analog channels) in NR1 format <validity> ::= {Cal Valid Cal Invalid}
:CHANnel<n>:PROBe:EXTernal {{0 OFF} {1 ON}} (see page 360)	:CHANnel<n>:PROBe:EXTernal? (see page 360)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {0 1}
:CHANnel<n>:PROBe:EXTernal:GAIN <gain_factor> (see page 361)	:CHANnel<n>:PROBe:EXTernal:GAIN? (see page 361)	<n> ::= 1 to (# analog channels) in NR1 format <gain_factor> ::= a real number from 0.0001 to 1000 in NR3 format
:CHANnel<n>:PROBe:EXTernal:OFFSet <offset_value> (see page 362)	:CHANnel<n>:PROBe:EXTernal:OFFSet? (see page 362)	<n> ::= 1 to (# analog channels) in NR1 format <offset_value> ::= a real number for the offset value at center screen in NR3 format
:CHANnel<n>:PROBe:EXTernal:UNITs <units> (see page 364)	:CHANnel<n>:PROBe:EXTernal:UNITs? (see page 364)	<n> ::= 1 to (# analog channels) in NR1 format <units> ::= {VOLT AMPere}
:CHANnel<n>:PROBe:HEAD[:TYPE] <head_param> (see page 365)	:CHANnel<n>:PROBe:HEAD[:TYPE]? (see page 365)	<head_param> ::= {SEND0 SEND6 SEND12 SEND20 DIFF0 DIFF6 DIFF12 DIFF20 DSMA DSMA6 NONE} <n> ::= 1 to (# analog channels) in NR1 format
n/a	:CHANnel<n>:PROBe:ID? (see page 366)	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:MODE <setting> (see page 367)	:CHANnel<n>:PROBe:MODE? (see page 367)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {DIFFerential DOFFset SEA SEB CM}
:CHANnel<n>:PROBe:RSENse <value> (see page 368)	:CHANnel<n>:PROBe:RSENse? (see page 368)	<value> ::= Ohms in NR3 format <n> ::= 1 to (# analog channels) in NR1 format

Table 8 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe:SKEW <skew_value> (see page 369)	:CHANnel<n>:PROBe:SKEW? (see page 369)	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1 to (# analog channels) in NR1 format
n/a	:CHANnel<n>:PROBe:STATUS? (see page 370)	<n> ::= 1 to (# analog channels) in NR1 format <status> ::= {<PS000XA_status> <N2893A_status>} <PS000XA_status> ::= {Laser Error Sensor Head Error Calibrating OK} <N2893A_status> ::= {Calibrating OK}
:CHANnel<n>:PROBe:SType <signal type> (see page 371)	:CHANnel<n>:PROBe:SType? (see page 371)	<signal type> ::= {DIFFerential SINGLE ISOLated} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:ZOOM {{0 OFF} {1 ON}} (see page 372)	:CHANnel<n>:PROBe:ZOOM? (see page 372)	<setting> ::= {0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROTection (see page 373)	:CHANnel<n>:PROTection? (see page 373)	{NORM TRIP} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:RANGe <range>[suffix] (see page 374)	:CHANnel<n>:RANGe? (see page 374)	<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 page 375)	:CHANnel<n>:SCALe? (see page 375)	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V mV} <n> ::= 1 to (# analog channels) in NR1 format

Table 8 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:UNITs <units> (see page 376)	:CHANnel<n>:UNITs? (see page 376)	<units> ::= {VOLT AMPere} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:VERNier { {0 OFF} {1 ON} } (see page 377)	:CHANnel<n>:VERNier? (see page 377)	{0 1} <n> ::= 1 to (# analog channels) in NR1 format

Table 9 :CLOCK Commands Summary

Command	Query	Options and Query Returns
:CLOCK { {0 OFF} {1 ON} } (see page 381)	:CLOCK? (see page 381)	{0 1}
:CLOCK:METHod <method> (see page 382)	:CLOCK:METHod? (see page 383)	<method> ::= {<fixed> <fopll> <sopll> <explicit>} <fixed> ::= FIXed, {AUTO SEMI[, <data_rate>] <data_rate>} <fopll> ::= FOPLL, <data_rate>, <loop_bandwidth> <sopll> ::= SOPLL, <data_rate>, <loop_bandwidth>, <damping_factor> <explicit> ::= EXPLicit, <source>, {RISing FALLing BOTH}[, <multiplier>] <data_rate> ::= the data signal's expected bits/second rate in NR3 format <loop_bandwidth> ::= the -3 dB bandwidth specification of PLL in NR3 format <damping_factor> ::= of the PLL's observed jitter transfer function in NR3 format <source> ::= explicit clock source, CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <multiplier> ::= explicit clock multiplier from 1 to 1000 in NR1 format

Table 10 :COUNter<c> Commands Summary

Command	Query	Options and Query Returns
n/a	:COUNter<c>:CURRent? (see page 387)	<c> ::= {1 2 3 4} <value> ::= current counter value in NR3 format
:COUNter<c>:ENABle { {0 OFF} {1 ON} } (see page 388)	:COUNter<c>:ENABle? (see page 388)	<c> ::= {1 2 3 4} {0 1}
:COUNter<c>:MODE <mode> (see page 389)	:COUNter<c>:MODE (see page 389)	<c> ::= {1 2 3 4} <mode> ::= {FREQuency PERiod TOTalize}
:COUNter<c>:NDIGits <value> (see page 390)	:COUNter<c>:NDIGits (see page 390)	<c> ::= {1 2 3 4} <value> ::= 3 to 8 in NR1 format
:COUNter<c>:SOURce <source> (see page 391)	:COUNter<c>:SOURce? (see page 391)	<c> ::= {1 2 3 4} <source> ::= {CHANnel<n> TQEvent} <n> ::= 1 to (# analog channels) in NR1 format
:COUNter<c>:TOTalize: CLear (see page 392)	n/a	<c> ::= {1 2 3 4}
:COUNter<c>:TOTalize: SLOPe <slope> (see page 393)	:COUNter<c>:TOTalize: SLOPe? (see page 393)	<c> ::= {1 2 3 4} <slope> ::= { {NEGative FALLing} {POSitive RISing} }

Table 11 :DIGital<d> Commands Summary

Command	Query	Options and Query Returns
:DIGital:ORDer:ASCend ing (see page 397)	n/a	n/a
:DIGital:ORDer:DESCen ding (see page 398)	n/a	n/a
:DIGital:ORDer[:SET] <digital_channels_lis t> (see page 399)	:DIGital:ORDer[:SET]? (see page 399)	<digital_channels_list> ::= comma-separated list, e.g., DIGital<d>,DIGital<d>,... <d> ::= 0 to (# digital channels - 1) in NR1 format
:DIGital<d>:DISPlay { {0 OFF} {1 ON} } (see page 400)	:DIGital<d>:DISPlay? (see page 400)	<d> ::= 0 to (# digital channels - 1) in NR1 format {0 1}

Table 11 :DIGital<d> Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGital<d>:LABel <string> (see page 401)	:DIGital<d>:LABel? (see page 401)	<d> ::= 0 to (# digital channels - 1) in NR1 format <string> ::= any series of 32 or less ASCII characters enclosed in quotation marks
n/a	:DIGital<d>:SETup? (see page 402)	<d> ::= 0 to (# digital channels - 1) in NR1 format Returns setup information string, for example, ":DIG0:DISP 1;THR +1.40E+00"
:DIGital<d>:SIZE <value> (see page 403)	:DIGital<d>:SIZE? (see page 403)	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {SMALl MEDium LARGe}
:DIGital<d>:THReshold <value>[suffix] (see page 404)	:DIGital<d>:THReshold ? (see page 404)	<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 12 :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n> > {{0 OFF} {1 ON}} (see page 410)	:DISPlay:ANNotation<n> >? (see page 410)	{0 1} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n> >:BACKground <mode> (see page 411)	:DISPlay:ANNotation<n> >:BACKground? (see page 411)	<mode> ::= {OPAQue INVerted TRANSPARENT} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n> >:COLor <color> (see page 412)	:DISPlay:ANNotation<n> >:COLor? (see page 412)	<color> ::= {CH1 CH2 CH3 CH4 DIG MATH REF MARKer WHITe RED} <n> ::= an integer from 1 to 20 in NR1 format.

Table 12 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n>:GRID <number> (see page 413)	:DISPlay:ANNotation<n>:GRID? (see page 413)	<number> ::= an integer from 1 to 20 in NR1 format. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:MODE <mode> (see page 414)	:DISPlay:ANNotation<n>:MODE? (see page 414)	<mode> ::= {GRID SOURce} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:SOURce <source> (see page 415)	:DISPlay:ANNotation<n>:SOURce? (see page 415)	<source> ::= {CHANnel<n> WMemory<n> FUNctIon<n> FFT DIGital<n>} <n> ::= an integer in NR1 format.
:DISPlay:ANNotation<n>:TEXT <string> (see page 416)	:DISPlay:ANNotation<n>:TEXT? (see page 416)	<string> ::= quoted ASCII string (up to 254 characters) <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:XPOSition <value> (see page 417)	:DISPlay:ANNotation<n>:XPOSition? (see page 417)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:YPOSition <value> (see page 418)	:DISPlay:ANNotation<n>:YPOSition? (see page 418)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:BACKlight {{0 OFF} {1 ON}} (see page 419)	n/a	n/a
:DISPlay:CLear (see page 420)	n/a	n/a
:DISPlay:CLOCK:IGRid {{0 OFF} {1 ON}} (see page 421)	:DISPlay:CLOCK:IGRid? (see page 421)	<setting> ::= {0 1}
:DISPlay:CLOCK:IGRid:XPOSition <value> (see page 422)	:DISPlay:CLOCK:IGRid:XPOSition? (see page 422)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.
:DISPlay:CLOCK:IGRid:YPOSition <value> (see page 423)	:DISPlay:CLOCK:IGRid:YPOSition? (see page 423)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.

Table 12 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:CLOCK[:STATe] [{0 OFF} {1 ON}] (see page 424)	:DISPlay:CLOCK[:STATe]? (see page 424)	<setting> ::= {0 1}
n/a	:DISPlay:DATA? [<format>] (see page 425)	<format> ::= {BMP PNG} <display data> ::= data in IEEE 488.2 # format
:DISPlay:DIALog:CLOSe (see page 426)	n/a	n/a
:DISPlay:DIALog:TRANs parent [{0 OFF} {1 ON}] (see page 427)	:DISPlay:DIALog:TRANs parent? (see page 427)	<setting> ::= {0 1}
:DISPlay:GRATicule:ALABels [{0 OFF} {1 ON}] (see page 428)	:DISPlay:GRATicule:ALABels? (see page 428)	<setting> ::= {0 1}
:DISPlay:GRATicule:ALABels:DUAL [{0 OFF} {1 ON}] (see page 429)	:DISPlay:GRATicule:ALABels:DUAL? (see page 429)	<setting> ::= {0 1}
:DISPlay:GRATicule:ALABels:IGRid [{0 OFF} {1 ON}] (see page 430)	:DISPlay:GRATicule:ALABels:IGRid? (see page 430)	<setting> ::= {0 1}
:DISPlay:GRATicule:COUnT <value> (see page 431)	:DISPlay:GRATicule:COUnT? (see page 431)	<num_grids> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:GRATicule:FScreen [{0 OFF} {1 ON}] (see page 432)	:DISPlay:GRATicule:FScreen? (see page 432)	<setting> ::= {0 1}
:DISPlay:GRATicule:INTensity <value> (see page 433)	:DISPlay:GRATicule:INTensity? (see page 433)	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:GRATicule:LAYout <type> (see page 434)	:DISPlay:GRATicule:LAYout? (see page 434)	<type> ::= {STACKed TILED OVERlay CUSTom}
:DISPlay:GRATicule:SEt <source>, <grid> (see page 435)	n/a	<source> ::= {CHANnel<n> WMemory<n> FUNCTION<n> FFT {PBUS<n> SBUS<n>} GAIN PHASe} <grid> ::= a grid number integer from 1 to 4 in NR1 format.

Table 12 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:DISPlay:GRATicule:SO URce? <grid> (see page 436)	<grid> ::= a grid number integer from 1 to 4 in NR1 format. Returns: <sources> ::= comma-separated list of the waveforms displayed in the grid. Can contain items like: CHAN<n>, WMEM<n>, FUNC<n>, FFT, SBUS<n>, GAIN, PHASe
:DISPlay:INTensity:WA Veform <value> (see page 437)	:DISPlay:INTensity:WA Veform? (see page 437)	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:LABel {{0 OFF} {1 ON}} (see page 438)	:DISPlay:LABel? (see page 438)	{0 1}
:DISPlay:LABList <binary block> (see page 439)	:DISPlay:LABList? (see page 439)	<binary block> ::= an ordered list of up to 75 labels, each 10 characters maximum, separated by newline characters
:DISPlay:MESSAge:CLEa r (see page 440)	n/a	n/a
:DISPlay:PERSiStence <value> (see page 441)	:DISPlay:PERSiStence? (see page 441)	<value> ::= {MINimum INFinite <time>} <time> ::= seconds in in NR3 format from 100E-3 to 60E0
:DISPlay:PERSiStence: CLEAr (see page 442)	n/a	n/a
n/a	:DISPlay:RESults:CATA log? (see page 443)	<windows> ::= list of the results windows displayed from left to right. Can contain: MEAS, MARK, HIST, DVM, COUN, LIST, MTES, FRAN
:DISPlay:RESults:LAYo ut <format> (see page 444)	:DISPlay:RESults:LAYo ut? (see page 444)	<format> ::= {TAB LIST}
:DISPlay:RESults:LAYo ut:LIST[:SElect] <results_window> (see page 445)	:DISPlay:RESults:LAYo ut:LIST[:SElect]? (see page 445)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}
:DISPlay:RESults:LAYo ut:TAB:LEFT[:SElect] <results_window> (see page 446)	:DISPlay:RESults:LAYo ut:TAB:LEFT[:SElect]? (see page 446)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}

Table 12 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] <results_window> (see page 447)	:DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] ? (see page 447)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}
:DISPlay:RESults:SIZE <size> (see page 448)	:DISPlay:RESults:SIZE ? (see page 448)	<size> ::= {CUSTom,<height> FULL HIDDEN} <height> ::= a pixel height integer from 36 to 1024 in NR1 format.
:DISPlay:TRANSPARENT {OFF ON} (see page 449)	:DISPlay:TRANSPARENT? (see page 449)	{OFF ON}
:DISPlay:VECTors {{0 OFF} {1 ON}} (see page 450)	:DISPlay:VECTors? (see page 450)	{0 1}
:DISPlay:XYColor <xy_color_scheme> (see page 451)	:DISPlay:XYColor? (see page 451)	<xy_color_scheme> ::= {VIRidis HOT ABYSS SKY INFerno}

Table 13 :DVM Commands Summary

Command	Query	Options and Query Returns
:DVM:ARANge {{0 OFF} {1 ON}} (see page 454)	:DVM:ARANge? (see page 454)	{0 1}
n/a	:DVM:CURREnt? (see page 455)	<dvm_value> ::= floating-point number in NR3 format
:DVM:ENABle {{0 OFF} {1 ON}} (see page 456)	:DVM:ENABle? (see page 456)	{0 1}
n/a	:DVM:FREQuency? (see page 455)	<freq_value> ::= floating-point number in NR3 format
:DVM:MODE <mode> (see page 457)	:DVM:MODE? (see page 457)	<dvm_mode> ::= {ACRMs DC DCRMs FREQuency}
:DVM:SOURce <source> (see page 458)	:DVM:SOURce? (see page 458)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format

Table 14 :EXtErnal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXtErnal:BWLimit <bwlimit> (see page 460)	:EXtErnal:BWLimit? (see page 460)	<bwlimit> ::= {0 OFF}
:EXtErnal:PROBe <attenuation> (see page 461)	:EXtErnal:PROBe? (see page 461)	<attenuation> ::= probe attenuation ratio in NR3 format
:EXtErnal:RANGe <range>[<suffix>] (see page 462)	:EXtErnal:RANGe? (see page 462)	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V mV}
:EXtErnal:UNITs <units> (see page 463)	:EXtErnal:UNITs? (see page 463)	<units> ::= {VOLT AMPere}

Table 15 :FFT Commands Summary

Command	Query	Options and Query Returns
:FFT:AVERage:COUNT <count> (see page 468)	:FFT:AVERage:COUNT? (see page 468)	<count> ::= an integer from 2 to 65536 in NR1 format.
n/a	:FFT:BSIZE? (see page 469)	<bin_size> ::= the FFT resolution as a bin size frequency width in NR3 format.
:FFT:CENTer <frequency> (see page 470)	:FFT:CENTer? (see page 470)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.
:FFT:CLear (see page 471)	n/a	n/a
:FFT:DETection:POINts <pts_per_span> (see page 472)	:FFT:DETection:POINts ? (see page 472)	<pts_per_span> ::= an integer from 640 to 65536 in NR1 format.
:FFT:DETection:TYPE <detection> (see page 473)	:FFT:DETection:TYPE? (see page 473)	<detection> ::= {OFF SAMPlE PPOSitive PNEGative AVERage NORMal}
:FFT:DISPlay {{0 OFF} {1 ON}} (see page 474)	:FFT:DISPlay? (see page 474)	<s> ::= 1-6, in NR1 format. {0 1}
:FFT:DMODE <display_mode> (see page 475)	:FFT:DMODE? (see page 475)	<display_mode> ::= {NORMal AVERage MAXHold MINHold}

Table 15 :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFT:FREQuency:START <frequency> (see page 476)	:FFT:FREQuency:START? (see page 476)	<frequency> ::= the start frequency in NR3 format.
:FFT:FREQuency:STOP <frequency> (see page 477)	:FFT:FREQuency:STOP? (see page 477)	<frequency> ::= the stop frequency in NR3 format.
:FFT:GATE <gating> (see page 478)	:FFT:GATE? (see page 478)	<gating> ::= {NONE ZOOM MARKers}
:FFT:OFFSet <offset> (see page 480)	:FFT:OFFSet? (see page 480)	<offset> ::= the value at center screen in NR3 format.
:FFT:RANGe <range> (see page 481)	:FFT:RANGe? (see page 481)	<range> ::= the full-scale vertical axis value in NR3 format.
n/a	:FFT:RBWidth? (see page 482)	<resolution_bw> ::= the FFT resolution as a resolution bandwidth frequency in NR3 format.
:FFT:READout <type> (see page 483)	:FFT:READout? (see page 483)	<type> ::= {OFF SRATe BSIZE RBWidth}
:FFT:REFeRence <level> (see page 484)	:FFT:REFeRence? (see page 484)	<level> ::= the current reference level in NR3 format.
:FFT:SCALe <scale value>[<suffix>] (see page 485)	:FFT:SCALe? (see page 485)	<scale_value> ::= integer in NR1 format. <suffix> ::= dB
:FFT:SOURce <source> (see page 486)	:FFT:SOURce? (see page 486)	<source> ::= {CHANnel<n> FUNCTion<m> MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format. <m> ::= 1 to (# math functions) in NR1 format
:FFT:SPAN (see page 487)	:FFT:SPAN? (see page 487)	 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.
n/a	:FFT:SRATe? (see page 488)	<sample_rate> ::= the FFT resolution as a sample rate frequency in NR3 format.
:FFT:VTYPe <units> (see page 489)	:FFT:VTYPe? (see page 489)	<units> ::= {DECibel VRMS}

Table 15 :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFT:WINDow <window> (see page 490)	:FFT:WINDow? (see page 490)	<window> ::= {RECTangular HANning FLATtop BHARRis BARTlett}
:FFTMax:OFFSet <offset>[<suffix>] (see page 492)	:FFTMax:OFFSet? (see page 492)	<offset> ::= the value at center screen in NR3 format. <suffix> ::= {dB mdB udB}
:FFTMax:RESet (see page 493)	n/a	n/a
:FFTMax:SCALe <scale_value>[<suffix>] >] (see page 494)	:FFTMax:SCALe? (see page 494)	<scale_value> ::= floating-point value in NR3 format. <suffix> ::= {dB mdB udB}
:FFTMax:STATe {{0 OFF} {1 ON}} (see page 495)	:FFTMax:STATe? (see page 495)	<setting> ::= {0 1}

Table 16 :FRANalysis Commands Summary

Command	Query	Options and Query Returns
n/a	:FRANalysis:DATA? [SWEep SINGLE] (see page 500)	<binary_block> ::= comma-separated data with newlines at the end of each row
:FRANalysis:ENABle {{0 OFF} {1 ON}} (see page 501)	:FRANalysis:ENABle? (see page 501)	{0 1}
:FRANalysis:FREQuency :MODE <setting> (see page 502)	:FRANalysis:FREQuency :MODE? (see page 502)	<setting> ::= {SWEep SINGLE}
:FRANalysis:FREQuency :SINGLE <value>[suffix] (see page 503)	:FRANalysis:FREQuency :SINGLE? (see page 503)	<value> ::= {20 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:FRANalysis:FREQuency :START <value>[suffix] (see page 504)	:FRANalysis:FREQuency :START? (see page 504)	<value> ::= {20 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:FRANalysis:FREQuency :STOP <value>[suffix] (see page 505)	:FRANalysis:FREQuency :STOP? (see page 505)	<value> ::= {100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}

Table 16 :FRANalysis Commands Summary (continued)

Command	Query	Options and Query Returns
:FRANalysis:PPDecade <value> (see page 506)	:FRANalysis:PPDecade? (see page 506)	<value> ::= {10 20 30 40 50 60 70 80 90 100}
:FRANalysis:RUN (see page 507)	n/a	n/a
:FRANalysis:SOURce:IN Put <source> (see page 508)	:FRANalysis:SOURce:IN Put? (see page 508)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:FRANalysis:SOURce:OU TPut <source> (see page 509)	:FRANalysis:SOURce:OU TPut? (see page 509)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:FRANalysis:SWEep:POI Nts <points> (see page 510)	:FRANalysis:SWEep:POI Nts? (see page 510)	<points> ::= integer from 1-1000 in NR1 format
:FRANalysis:TRACe <selection> (see page 511)	:FRANalysis:TRACe? (see page 511)	<selection> ::= {NONE ALL GAIN PHASE}[,{GAIN PHASE}]
:FRANalysis:WGEN:LOAD <impedance> (see page 512)	:FRANalysis:WGEN:LOAD ? (see page 512)	<impedance> ::= {ONEMeg FIFTy}
:FRANalysis:WGEN:VOLT age <amplitude>,[<range>] (see page 513)	:FRANalysis:WGEN:VOLT age? [<range>] (see page 513)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F20HZ F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ F20MHZ}
:FRANalysis:WGEN:VOLT age:PROFile {{0 OFF} {1 ON}} (see page 515)	:FRANalysis:WGEN:VOLT age:PROFile? (see page 515)	{0 1}

Table 17 :FUNCTion<m> Commands Summary

Command	Query	Options and Query Returns
:FUNCTion<m>:AUTosetu p (see page 524)	n/a	n/a
:FUNCTion<m>:AUTosetu p:ENABle {{0 OFF} {1 ON}} (see page 525)	:FUNCTion<m>:AUTosetu p:ENABle? (see page 525)	<m> ::= 1 to (# math functions) in NR1 format <setting> ::= {0 1}

Table 17 :FUNctioN<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNctioN<m>:AVERAge:COUNT <count> (see page 526)	:FUNctioN<m>:AVERAge:COUNT? (see page 526)	<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 527)	:FUNctioN<m>:BUS:CLOCK? (see page 527)	<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 528)	:FUNctioN<m>:BUS:SLOPe? (see page 528)	<slope> ::= {NEGative POSitive EITHer} <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>:BUS:YINCrement <value> (see page 529)	:FUNctioN<m>:BUS:YINCrement? (see page 529)	<value> ::= value per bus code, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>:BUS:YORigin <value> (see page 530)	:FUNctioN<m>:BUS:YORigin? (see page 530)	<value> ::= value at bus code = 0, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>:BUS:YUNits <units> (see page 531)	:FUNctioN<m>:BUS:YUNits? (see page 531)	<units> ::= {VOLT AMPere NONE} <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>:CLEAr (see page 532)	n/a	n/a
:FUNctioN<m>:DISPlay {{0 OFF} {1 ON}} (see page 533)	:FUNctioN<m>:DISPlay? (see page 533)	{0 1} <m> ::= 1 to (# math functions) in NR1 format
n/a	:FUNctioN<m>[:FFT]:BSIZE? (see page 534)	<bin_size> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format

Table 17 :FUNctioN<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNctioN<m>[:FFT]:CENTer <frequency> (see page 535)	:FUNctioN<m>[:FFT]:CENTer? (see page 535)	<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]:DETEction:POINTs <number_of_buckets> (see page 536)	:FUNctioN<m>[:FFT]:DETEction:POINTs? (see page 536)	<number_of_buckets> ::= an integer in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>[:FFT]:DETEction:TYPE <type> (see page 537)	:FUNctioN<m>[:FFT]:DETEction:TYPE? (see page 537)	<type> ::= {OFF SAMPlE PPOSitive PNENegative NORMAl AVERAge} <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>[:FFT]:FREQuency:START <frequency> (see page 538)	:FUNctioN<m>[:FFT]:FREQuency:START? (see page 538)	<frequency> ::= the start frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>[:FFT]:FREQuency:STOP <frequency> (see page 539)	:FUNctioN<m>[:FFT]:FREQuency:STOP? (see page 539)	<frequency> ::= the stop frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>:FFT:GATE <gating> (see page 540)	:FUNctioN<m>:FFT:GATE? (see page 540)	<gating> ::= {NONE ZOOM MARKers} <m> ::= 1-4 in NR1 format
:FUNctioN<m>[:FFT]:PHASe:REfERence <ref_point> (see page 542)	:FUNctioN<m>[:FFT]:PHASe:REfERence? (see page 542)	<ref_point> ::= {TRIGger DISPlay} <m> ::= 1-4 in NR1 format
n/a	:FUNctioN<m>[:FFT]:RBWidth? (see page 543)	<resolution_bw> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctioN<m>[:FFT]:READout<n> <readout_type> (see page 544)	:FUNctioN<m>[:FFT]:READout<n>? (see page 544)	<readout_type> ::= {SRATe BSIZE RBWidth} <m> ::= 1 to (# math functions) in NR1 format <n> ::= 1-2 in NR1 format, 2 is for dedicated FFT function

Table 17 :FUNction<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNction<m>[:FFT]:SPAN (see page 545)	:FUNction<m>[:FFT]:SPAN? (see page 545)	 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz. <m> ::= 1 to (# math functions) in NR1 format
n/a	:FUNction<m>[:FFT]:SRATE? (see page 546)	<sample_rate> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>[:FFT]:VTYPe <units> (see page 547)	:FUNction<m>[:FFT]:VTYPe? (see page 547)	<units> ::= {DECibel VRMS} for the FFT (magnitude) operation <units> ::= {DEGREes RADians} for the FFTPhase operation <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>[:FFT]:WINDow <window> (see page 548)	:FUNction<m>[:FFT]:WINDow? (see page 548)	<window> ::= {RECTangular HANNing FLATtop BHARRis BARTlett} <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:FREQuency:BANDpass:CENTer <center_freq> (see page 549)	:FUNction<m>:FREQuency:BANDpass:CENTer? (see page 549)	<center_freq> ::= center frequency of band-pass filter in NR3 format
:FUNction<m>:FREQuency:BANDpass:WIDTh <freq_width> (see page 550)	:FUNction<m>:FREQuency:BANDpass:WIDTh? (see page 550)	<freq_width> ::= frequency width of band-pass filter in NR3 format
:FUNction<m>:FREQuency:HIGHPass <3dB_freq> (see page 551)	:FUNction<m>:FREQuency:HIGHPass? (see page 551)	<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 552)	:FUNction<m>:FREQuency:LOWPass? (see page 552)	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:GATE <gating> (see page 553)	:FUNction<m>:GATE? (see page 554)	<gating> ::= {NONE ZOOM BOTH MARKers} <m> ::= 1-4 in NR1 format

Table 17 :FUNctio<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNctio<m>:INTEGRate:IICondition {{0 OFF} {1 ON}} (see page 555)	:FUNctio<m>:INTEGRate:IICondition? (see page 555)	{0 1} <m> ::= 1 to (# math functions) in NR1 format
:FUNctio<m>:INTEGRate:IOFFset <input_offset> (see page 556)	:FUNctio<m>:INTEGRate:IOFFset? (see page 556)	<input_offset> ::= DC offset correction in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNctio<m>:LABel <string> (see page 557)	:FUNctio<m>:LABel? (see page 557)	<string> ::= quoted ASCII string
:FUNctio<m>:LINear:GAIN <value> (see page 558)	:FUNctio<m>:LINear:GAIN? (see page 558)	<value> ::= 'A' in $Ax + B$, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctio<m>:LINear:OFFSet <value> (see page 559)	:FUNctio<m>:LINear:OFFSet? (see page 559)	<value> ::= 'B' in $Ax + B$, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNctio<m>:OFFSet <offset> (see page 560)	:FUNctio<m>:OFFSet? (see page 560)	<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
:FUNctio<m>:OPERatioN <operation> (see page 561)	:FUNctio<m>:OPERatioN? (see page 563)	<operation> ::= {ADD SUBTract MULTiply DIVide INTEGRate DIFF FFT FFTPhase SQRT MAGNify ABSolute SQUare LN LOG EXP TEN LOWPass HIGHpass BANDpass AVERage LINear MAXimum MINimum PEAK MAXHold MINHold TREND} <m> ::= 1 to (# math functions) in NR1 format

Table 17 :FUNction<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNction<m>:RANGe <range> (see page 565)	:FUNction<m>:RANGe? (see page 565)	<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>:REFerence <level> (see page 566)	:FUNction<m>:REFerence? (see page 566)	<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
:FUNction<m>:SCALE <scale value>[<suffix>] (see page 567)	:FUNction<m>:SCALE? (see page 567)	<scale value> ::= integer in NR1 format <suffix> ::= {V dB} <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:SMOoth:POINTs <points> (see page 568)	:FUNction<m>:SMOoth:POINTs? (see page 568)	<points> ::= odd integer in NR1 format

Table 17 :FUNction<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNction<m>:SOURce1 <source> (see page 569)	:FUNction<m>:SOURce1? (see page 569)	<source> ::= {CHANnel<n> FUNction<c> MATH<c> WMEMory<r> BUS} <n> ::= 1 to (# analog channels) in NR1 format <c> ::= {1 2 3}, must be lower than <m> <r> ::= 1 to (# ref waveforms) in NR1 format ::= {1 2} <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:SOURce2 <source> (see page 571)	:FUNction<m>:SOURce2? (see page 571)	<source> ::= {CHANnel<n> WMEMory<r> NONE} <n> ::= 1 to (# analog channels) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:TREND:NMEASUREMENT MEAS<n> (see page 572)	:FUNction<m>:TREND:NMEASUREMENT? (see page 572)	<n> ::= # of installed measurement, from 1 to 8 <m> ::= 1 to (# math functions) in NR1 format

Table 18 :HCOpy Commands Summary

Command	Query	Options and Query Returns
n/a	:HCOpy:SDUMp:DATA? (see page 574)	<display_data> ::= binary block data in IEEE-488.2 # format.
:HCOpy:SDUMp[:DATA]:FORMMat <format> (see page 575)	:HCOpy:SDUMp[:DATA]:FORMMat? (see page 575)	<format> ::= {BMP PNG}

Table 19 :HISTogram Commands Summary

Command	Query	Options and Query Returns
:HISTogram:AXIS <axis> (see page 580)	:HISTogram:AXIS? (see page 580)	<axis> ::= {VERTical HORizontal}
:HISTogram:DISPlay { {0 OFF} {1 ON} } (see page 581)	:HISTogram:DISPlay? (see page 581)	{0 1}
:HISTogram:MEASurement MEAS<n> (see page 582)	:HISTogram:MEASurement? (see page 582)	<n> ::= # of installed measurement, from 1 to 10
:HISTogram:MODE <mode> (see page 583)	:HISTogram:MODE? (see page 583)	<mode> ::= {OFF WAVEform MEASurement}
:HISTogram:RESet (see page 584)	n/a	n/a
:HISTogram:SIZE <size> (see page 585)	:HISTogram:SIZE? (see page 585)	<size> ::= 1.0 to 5.0 for vertical histograms, 1.0 to 4.0 for horizontal histograms, in NR3 format
:HISTogram:TYPE <type> (see page 586)	:HISTogram:TYPE? (see page 586)	<type> ::= {VERTical HORizontal MEASurement}
:HISTogram:WINDow:BLIMit <limit> (see page 587)	:HISTogram:WINDow:BLIMit? (see page 587)	<limit> ::= bottom value in NR3 format
:HISTogram:WINDow:LLIMit <limit> (see page 588)	:HISTogram:WINDow:LLIMit? (see page 588)	<limit> ::= left value in NR3 format
:HISTogram:WINDow:RLIMit <limit> (see page 589)	:HISTogram:WINDow:RLIMit? (see page 589)	<limit> ::= right value in NR3 format
:HISTogram:WINDow:SOURce <source> (see page 590)	:HISTogram:WINDow:SOURce? (see page 590)	<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
:HISTogram:WINDow:TLLIMit <limit> (see page 591)	:HISTogram:WINDow:TLLIMit? (see page 591)	<limit> ::= top value in NR3 format

Table 20 :JITter Commands Summary

Command	Query	Options and Query Returns
:JITter:HISTogram:APPLy (see page 595)	n/a	n/a
:JITter:MEASurement MEAS<n> (see page 596)	:JITter:MEASurement? (see page 596)	<n> ::= # of installed measurement, from 1 to 10
:JITter:SPECTrum:APPLy (see page 598)	n/a	n/a
:JITter:SPECTrum:HORizontal:POSition <value> (see page 599)	:JITter:SPECTrum:HORizontal:POSition? (see page 599)	<value> ::= X position (center), in NR3 format
:JITter:SPECTrum:HORizontal:RANGe <value> (see page 600)	:JITter:SPECTrum:HORizontal:RANGe? (see page 600)	<value> ::= X scaling (span), in NR3 format
:JITter:SPECTrum:VERTical:OFFSet <value> (see page 601)	:JITter:SPECTrum:VERTical:OFFSet? (see page 601)	<value> ::= Y offset, in NR3 format
:JITter:SPECTrum:VERTical:RANGe <value> (see page 602)	:JITter:SPECTrum:VERTical:RANGe? (see page 602)	<value> ::= Y scaling, in NR3 format
:JITter:SPECTrum:VERTical:TYPE <type> (see page 603)	:JITter:SPECTrum:VERTical:TYPE? (see page 603)	<type> ::= {DECibel VRMS}
:JITter:SPECTrum:WINDow <window> (see page 604)	:JITter:SPECTrum:WINDow? (see page 604)	<window> ::= {RECTangular HANNing FLATtop BHARRis BARTlett}
:JITter:STATe {{0 OFF} {1 ON}} (see page 606)	:JITter:STATe? (see page 606)	{0 1}
:JITter:TREND:APPLy (see page 608)	n/a	n/a
:JITter:TREND:SMOoth:APPLy (see page 609)	n/a	n/a
:JITter:TREND:SMOoth:POINTs <points> (see page 610)	:JITter:TREND:SMOoth:POINTs? (see page 610)	<points> ::= odd integer in NR1 format
:JITter:TREND:VERTical:OFFSet <value> (see page 611)	:JITter:TREND:VERTical:OFFSet? (see page 611)	<value> ::= Y offset, in NR3 format
:JITter:TREND:VERTical:RANGe <value> (see page 612)	:JITter:TREND:VERTical:RANGe? (see page 612)	<value> ::= Y scaling, in NR3 format

Table 21 :LISTer Commands Summary

Command	Query	Options and Query Returns
n/a	:LISTer:DATA? (see page 614)	<binary_block> ::= comma-separated data with newlines at the end of each row
:LISTer:DISPlay { {OFF 0} {PBUS1 SBUS1 ON 1} {PBUS2 SBUS2 2} ALL } (see page 615)	:LISTer:DISPlay? (see page 615)	{OFF PBUS1 PBUS2 ALL}
:LISTer:REFeRence <time_ref> (see page 616)	:LISTer:REFeRence? (see page 616)	<time_ref> ::= {TRIGger PREVIOUS}

Table 22 :LTEST Commands Summary

Command	Query	Options and Query Returns
:LTEST:COpy (see page 619)	n/a	n/a
:LTEST:COpy:ALL (see page 620)	n/a	n/a
:LTEST:COpy:MARGin <percent> (see page 621)	:LTEST:COpy:MARGin? (see page 621)	<percent> ::= copy margin percent in NR1 format
:LTEST:ENABle { {0 OFF} {1 ON} } (see page 622)	:LTEST:ENABle? (see page 622)	<setting> ::= {0 1}
:LTEST:FAIL <condition> (see page 623)	:LTEST:FAIL? (see page 623)	<condition> ::= {INSide OUTSide}
:LTEST:LLIMit <lower_value> (see page 624)	:LTEST:LLIMit? (see page 624)	<lower_value> ::= a real number in NR3 format
:LTEST:MEASurement {MEAS<n>} (see page 625)	:LTEST:MEASurement? (see page 625)	<n> ::= # of installed measurement, from 1 to 10
n/a	:LTEST:RESults? [{MEAS<n>}] (see page 626)	<n> ::= # of installed measurement, from 1 to 10
:LTEST:RUMode:SOFailu re { {0 OFF} {1 ON} } (see page 627)	:LTEST:RUMode:SOFailu re? (see page 627)	<setting> ::= {0 1}

Table 22 :LTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:LTEST:TEST {{0 OFF} {1 ON}} (see page 628)	:LTEST:TEST? (see page 628)	<setting> ::= {0 1}
:LTEST:ULIMit <upper_value> (see page 629)	:LTEST:ULIMit? (see page 629)	<upper_value> ::= a real number in NR3 format

Table 23 :MARKer Commands Summary

Command	Query	Options and Query Returns
n/a	:MARKer:DYDX? (see page 634)	<return_value> ::= $\Delta Y / \Delta X$ value in NR3 format
:MARKer:MODE <mode> (see page 635)	:MARKer:MODE? (see page 635)	<mode> ::= {OFF MEASurement MANual WAVEform BINary HEX}
:MARKer:X1:DISPlay {{0 OFF} {1 ON}} (see page 636)	:MARKer:X1:DISPlay? (see page 636)	<setting> ::= {0 1}
:MARKer:X1Position <position>[suffix] (see page 637)	:MARKer:X1Position? (see page 637)	<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 page 638)	:MARKer:X1Y1source? (see page 638)	<source> ::= {CHANnel<n> FUNCTION<m> FFT MATH<m> WMemory<r> HISTogram} <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 page 639)	:MARKer:X2:DISPlay? (see page 639)	<setting> ::= {0 1}

Table 23 :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:X2Position <position>[suffix] (see page 640)	:MARKer:X2Position? (see page 640)	<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 page 641)	:MARKer:X2Y2source? (see page 641)	<source> ::= {CHANnel<n> FUNCTION<m> FFT MATH<m> WMemory<r> HISTogram} <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 page 642)	<return_value> ::= X cursors delta value in NR3 format
:MARKer:XUNits <mode> (see page 643)	:MARKer:XUNits? (see page 643)	<units> ::= {SEConds HERTz DEGREes PERCent}
:MARKer:XUNits:USE (see page 644)	n/a	n/a
:MARKer:Y1:DISPlay {{0 OFF} {1 ON}} (see page 645)	:MARKer:Y1:DISPlay? (see page 645)	<setting> ::= {0 1}
:MARKer:Y1Position <position>[suffix] (see page 646)	:MARKer:Y1Position? (see page 646)	<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 page 647)	:MARKer:Y2:DISPlay? (see page 647)	<setting> ::= {0 1}
:MARKer:Y2Position <position>[suffix] (see page 648)	:MARKer:Y2Position? (see page 648)	<position> ::= Y2 cursor position value in NR3 format [suffix] ::= {V mV dB} <return_value> ::= Y2 cursor position value in NR3 format

Table 23 :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MARKer:YDELta? (see page 649)	<return_value> ::= Y cursors delta value in NR3 format
:MARKer:YUNits <mode> (see page 650)	:MARKer:YUNits? (see page 650)	<units> ::= {BASE PERCent}
:MARKer:YUNits:USE (see page 651)	n/a	n/a

Table 24 :MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:AREa [<interval>] [, <source>] (see page 673)	:MEASure:AREa? [<interval>] [, <source>] (see page 673)	<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
:MEASure:BRATe [<source>] (see page 674)	:MEASure:BRATe? [<source>] (see page 674)	<source> ::= {<digital channels> CHANnel<n> FUNCTION<m> MATH<m> WMemory<r>} <digital channels> ::= DIGital<d> <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 <return_value> ::= bit rate in Hz, NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:BWIDth [<source>] (see page 675)	:MEASure:BWIDth? [<source>] (see page 675)	<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 page 676)	n/a	n/a
:MEASure:COUNter [<source>] (see page 677)	:MEASure:COUNter? [<source>] (see page 677)	<source> ::= {CHANnel<n> DIGital<d> EXTErnal} <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:CTCNwidth [<source>] (see page 679)	:MEASure:CTCNwidth? [<source>] (see page 679)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= -Width to -Width jitter in NR3 format
:MEASure:CTCPwidth [<source>] (see page 681)	:MEASure:CTCPwidth? [<source>] (see page 681)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= +Width to +Width jitter in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DElay [<source1>] [,<source2>] (see page 683)	:MEASure:DElay? [<source1>] [,<source2>] (see page 683)	<source1,2> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DElay:DEFine <source1_edge_slope>, <source1_edge_number> , <source1_edge_thresho ld>, <source2_edge_slope>, <source2_edge_number> , <source2_edge_thresho ld> (see page 685)	:MEASure:DElay:DEFine ? (see page 685)	<source1_edge_slope>, <source2_edge_slope> ::= {RISing FALLing} <source1_edge_number>, <source2_edge_number> ::= 0 to 1000 in NR1 format <source1_edge_threshold>, <source2_edge_threshold> ::= {LOWer MIDDle UPPer}
:MEASure:DUTYcycle [<source>] (see page 687)	:MEASure:DUTYcycle? [<source>] (see page 687)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FALLtime [<source>] (see page 688)	:MEASure:FALLtime? [<source>] (see page 688)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= time in seconds between the lower and upper thresholds in NR3 format
:MEASure:FFT:ACPR <chan_width>, <chan_spacing>, <chan>[,<source>] (see page 689)	:MEASure:FFT:ACPR? <chan_width>, <chan_spacing>, <chan>[,<source>] (see page 689)	<chan_width> ::= width of main range and sideband channels, Hz in NR3 format <chan_spacing> ::= spacing between main range and sideband channels, Hz in NR3 format <chan> ::= {CENTer HIGH<sb> LOW<sb>} <sb> ::= sideband 1 to 5 <source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= adjacent channel power ratio, dBV in NR3 format
:MEASure:FFT:CPOWer [<source>] (see page 690)	:MEASure:FFT:CPOWer? [<source>] (see page 690)	<source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= spectral channel power, dBV in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FFT:OBW <percentage>[,<source>] (see page 691)	:MEASure:FFT:OBW? <percentage>[,<source>] (see page 691)	<percentage> ::= percent of spectral power occupied bandwidth is measured for in NR3 format <source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= occupied bandwidth, Hz in NR3 format
:MEASure:FFT:THD [<source>] (see page 692)	:MEASure:FFT:THD? [<source>] (see page 692)	<source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= total harmonic distortion ratio percent in NR3 format
:MEASure:FREquency [<source>] (see page 693)	:MEASure:FREquency? [<source>] (see page 693)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= frequency in Hertz in NR3 format
n/a	:MEASure:HISTogram:BWIDth? (see page 694)	<value> ::= bin width in NR3 format
n/a	:MEASure:HISTogram:HITS? (see page 695)	<value> ::= # of total hits in NR3 format
n/a	:MEASure:HISTogram:M1S? (see page 696)	<value> ::= % of hits within +/-1 std dev in NR3 format
n/a	:MEASure:HISTogram:M2S? (see page 697)	<value> ::= % of hits within +/-2 std devs in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:HISTogram:M3S? (see page 698)	<value> ::= % of hits within +/-3 std devs in NR3 format
n/a	:MEASure:HISTogram:MAXimum? (see page 699)	<value> ::= value of max bin in NR3 format
n/a	:MEASure:HISTogram:MEAN? (see page 700)	<value> ::= mean of histogram in NR3 format
n/a	:MEASure:HISTogram:MEDian? (see page 701)	<value> ::= median of histogram in NR3 format
n/a	:MEASure:HISTogram:MINimum? (see page 702)	<value> ::= value of min bin in NR3 format
n/a	:MEASure:HISTogram:MODE? (see page 703)	<value> ::= mode of histogram in NR3 format
n/a	:MEASure:HISTogram:PEAK? (see page 704)	<value> ::= # of hits in max bin in NR3 format
n/a	:MEASure:HISTogram:PEAK? (see page 705)	<value> ::= delta value between max bin and min bin in NR3 format
n/a	:MEASure:HISTogram:SD EViation? (see page 706)	<value> ::= standard deviation of histogram in NR3 format
:MEASure:HYSTeresis { {0 OFF} {1 ON} } (see page 707)	:MEASure:HYSTeresis? (see page 707)	{ 0 1 }
:MEASure:NCJitter <config> (see page 708)	:MEASure:NCJitter? <config> (see page 709)	<config> ::= <source>, <direction>, <num_periods> <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 <direction> ::= {RISing FALLing} <num_periods> ::= 1 to 10000 in NR1 format <return_value> ::= N-Period to N-Period jitter in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NDUTy [<source>] (see page 710)	:MEASure:NDUTy? [<source>] (see page 710)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= ratio of negative pulse width to period in NR3 format
:MEASure:NEDGes [<source>] (see page 711)	:MEASure:NEDGes? [<source>] (see page 711)	<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 falling edge count in NR3 format
:MEASure:NPERiod <config> (see page 712)	:MEASure:NPERiod? <config> (see page 712)	<config> ::= <source>, <slope>, <num_periods> <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 <slope> ::= {RISing FALLing} <num_periods> ::= 1 to 10000 in NR1 format <return_value> ::= N-Period jitter in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NPULses [<source>] (see page 713)	:MEASure:NPULses? [<source>] (see page 713)	<source> ::= {CHANnel<n> FUNction<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= the falling pulse count in NR3 format
:MEASure:NWIDth [<source>] (see page 714)	:MEASure:NWIDth? [<source>] (see page 714)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>] (see page 715)	:MEASure:OVERshoot? [<source>] (see page 715)	<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 percent of the overshoot of the selected waveform in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PEDGes [<source>] (see page 717)	:MEASure:PEDGes? [<source>] (see page 717)	<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 rising edge count in NR3 format
:MEASure:PERiod [<source>] (see page 718)	:MEASure:PERiod? [<source>] (see page 718)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source1>] [,<source2>] (see page 719)	:MEASure:PHASe? [<source1>] [,<source2>] (see page 719)	<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> ::= the phase angle value in degrees in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PPULses [<source>] (see page 720)	:MEASure:PPULses? [<source>] (see page 720)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= the rising pulse count in NR3 format
:MEASure:PREShoot [<source>] (see page 721)	:MEASure:PREShoot? [<source>] (see page 721)	<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 percent of preshoot of the selected waveform in NR3 format
:MEASure:PWIDth [<source>] (see page 722)	:MEASure:PWIDth? [<source>] (see page 722)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= width of positive pulse in seconds in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:QUICK <source> (see page 723)	n/a	<source> ::= {CHANnel<n>}
n/a	:MEASure:RESults? <result_list> (see page 724)	<result_list> ::= comma-separated list of measurement results
:MEASure:RISetime [<source>] (see page 728)	:MEASure:RISetime? [<source>] (see page 728)	<source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEMemory<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> ::= rise time in seconds in NR3 format
:MEASure:SDEviation [<source>] (see page 729)	:MEASure:SDEviation? [<source>] (see page 729)	<source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEMemory<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 std deviation in NR3 format
:MEASure:SHOW {{0 OFF} {1 ON}} (see page 730)	:MEASure:SHOW? (see page 730)	{0 1}
:MEASure:SLEWrate [<source>[,<slope>]] (see page 731)	:MEASure:SLEWrate? [<source>[,<slope>]] (see page 731)	<source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEMemory<r>} <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

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:SOURce <source1> [,<source2>] (see page 733)	:MEASure:SOURce? (see page 733)	<source1,2> ::= {CHANnel<n> DIGital<d> FUNctIon<m> MATH<m> WMEMory<r> EXTernal} <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> ::= {<source> NONE}
:MEASure:STATistics <type> (see page 736)	:MEASure:STATistics? (see page 736)	<type> ::= {{ON 1} CURRent MEAN MINimum MAXimum STDDev COUNT} ON ::= all statistics returned
:MEASure:STATistics:D ISPlay {{0 OFF} {1 ON}} (see page 737)	:MEASure:STATistics:D ISPlay? (see page 737)	{0 1}
:MEASure:STATistics:I NCRement (see page 738)	n/a	n/a
:MEASure:STATistics:M COunt <setting> (see page 739)	:MEASure:STATistics:M COunt? (see page 739)	<setting> ::= {INFinite <count>} <count> ::= 2 to 2000 in NR1 format
:MEASure:STATistics:R ESet (see page 740)	n/a	n/a
:MEASure:STATistics:R SDeviation {{0 OFF} {1 ON}} (see page 741)	:MEASure:STATistics:R SDeviation? (see page 741)	{0 1}

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TEDGe [<slope>,<occurrence>,<source>] (see page 742)	:MEASure:TEDGe? [<slope>,<occurrence>,<source>] (see page 743)	<slope> ::= {RISing FALLing BOTH} <occurrence> ::= [+ -]<number> <number> ::= the edge number in NR1 format <source> ::= {<digital channels> CHANnel<n> FUNction<m> MATH<m> WMEMory<r>} <digital channels> ::= DIGital<d> <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 <return_value> ::= time in seconds of the specified transition
:MEASure:THResholds:ABSolute <upper>,<middle>,<lower> (see page 746)	:MEASure:THResholds:ABSolute? (see page 746)	<upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.
:MEASure:THResholds:METHod <threshold_mode> (see page 747)	:MEASure:THResholds:METHod? (see page 747)	<threshold_mode> ::= {PERCent ABSolute}
:MEASure:THResholds:PERCent <upper>,<middle>,<lower> (see page 748)	:MEASure:THResholds:PERCent? (see page 748)	<upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.
:MEASure:THResholds:STANdard (see page 749)	n/a	n/a

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TIEClock <config> (see page 750)	:MEASure:TIEClock? <config> (see page 750)	<config> ::= <source>, {SECond UNITinterval}, <direction> <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 <direction> ::= {RISing FALLing BOTH} <return_value> ::= Clock TIE jitter in NR3 format
:MEASure:TIEData <config> (see page 752)	:MEASure:TIEData? <config> (see page 752)	<config> ::= <source>, {SECond UNITinterval} <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> ::= Clock TIE jitter in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:TVALue? <value>, [<slope>]<occurrence> [,<source>] (see page 754)	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <source> ::= {CHANnel<n> DIGital<d> 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 <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 756)	:MEASure:VAMPlitude? [<source>] (see page 756)	<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 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAVerage [<interval>] [, <source>] (see page 757)	:MEASure:VAVerage? [<interval>] [, <source>] (see page 757)	<interval> ::= {CYCLe DISPlay} <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 <return_value> ::= calculated average voltage in NR3 format
:MEASure:VBASe [<source>] (see page 758)	:MEASure:VBASe? [<source>] (see page 758)	<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 <base_voltage> ::= voltage at the base of the selected waveform in NR3 format
:MEASure:VMAX [<source>] (see page 759)	:MEASure:VMAX? [<source>] (see page 759)	<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> ::= maximum voltage of the selected waveform in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VMIN [<source>] (see page 760)	:MEASure:VMIN? [<source>] (see page 760)	<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> ::= minimum voltage of the selected waveform in NR3 format
:MEASure:VPP [<source>] (see page 761)	:MEASure:VPP? [<source>] (see page 761)	<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> ::= voltage peak-to-peak of the selected waveform in NR3 format
:MEASure:VRATio [<interval>] [,<source 1>] [,<source2>] (see page 762)	:MEASure:VRATio? [<interval>] [,<source 1>] [,<source2>] (see page 762)	<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 <return_value> ::= the ratio value in dB in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VRMS [<interval>] [,<type>] [,<source>] (see page 763)	:MEASure:VRMS? [<interval>] [,<type>] [,<source>] (see page 763)	<interval> ::= {CYCLe DISPlay} <type> ::= {AC DC} <source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEMemory<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
:MEASure:VTOP [<source>] (see page 764)	:MEASure:VTOP? [<source>] (see page 764)	<source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEMemory<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 page 765)	:MEASure:WINDow? (see page 765)	<type> ::= {MAIN ZOOM AUTO GATE}
:MEASure:XMAX [<source>] (see page 766)	:MEASure:XMAX? [<source>] (see page 766)	<source> ::= {CHANnel<n> FUNCTION<m> FFT MATH<m> WMEMemory<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> ::= horizontal value of the maximum in NR3 format

Table 24 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:XMIN [<source>] (see page 767)	:MEASure:XMIN? [<source>] (see page 767)	<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> ::= horizontal value of the minimum in NR3 format
:MEASure:YATX <horiz_location>[,<source>] (see page 768)	:MEASure:YATX? <horiz_location>[,<source>] (see page 768)	<horiz_location> ::= displayed time from trigger in seconds in NR3 format <source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= voltage at the specified time in NR3 format

Table 25 :MEASure Power Commands Summary

Command	Query	Options and Query Returns
:MEASure:ANGLE [<source1>] [, <source2>] >] (see page 774)	:MEASure:ANGLE? [<source1>] [, <source2>] >] (see page 774)	<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 page 775)	:MEASure:APParent? [<source1>] [, <source2>] >] (see page 775)	<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 page 776)	:MEASure:CPLoss? [<source1>] [, <source2>] >] (see page 776)	<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 777)	:MEASure:CRESt? [<source>] (see page 777)	<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 778)	:MEASure:EFFiciency? (see page 778)	<return_value> ::= percent value in NR3 format

Table 25 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:ELOSS [<source>] (see page 779)	:MEASure:ELOSS? [<source>] (see page 779)	<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 780)	:MEASure:FACTOR? [<source1>] [, <source2>] >] (see page 780)	<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 781)	:MEASure:IPower? (see page 781)	<return_value> ::= the input power value in NR3 format
:MEASure:OFFTime [<source1>] [, <source2>] >] (see page 782)	:MEASure:OFFTime? [<source1>] [, <source2>] >] (see page 782)	<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 783)	:MEASure:ONTime? [<source1>] [, <source2>] >] (see page 783)	<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 784)	:MEASure:OPower? (see page 784)	<return_value> ::= the output power value in NR3 format

Table 25 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PCURrent [<source>] (see page 785)	:MEASure:PCURrent? [<source>] (see page 785)	<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 page 786)	:MEASure:PLOSs? [<source>] (see page 786)	<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
:MEASure:RDSON [<source1>] [,<source2 >] (see page 787)	:MEASure:RDSON? [<source1>] [,<source2 >] (see page 787)	<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 Rds(on) value in NR3 format
:MEASure:REACTive [<source1>] [,<source2 >] (see page 788)	:MEASure:REACTive? [<source1>] [,<source2 >] (see page 788)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the reactive power value in NR3 format

Table 25 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:REAL [<source>] (see page 789)	:MEASure:REAL? [<source>] (see page 789)	<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 790)	:MEASure:RIPPlE? [<source>] (see page 790)	<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 25 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TRESponse [<source>] (see page 791)	:MEASure:TRESponse? [<source>] (see page 791)	<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 792)	:MEASure:VCEsat? [<source>] (see page 792)	<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

Table 26 :MTESt Commands Summary

Command	Query	Options and Query Returns
:MTESt:ALL { {0 OFF} {1 ON} } (see page 799)	:MTESt:ALL? (see page 799)	{0 1}
:MTESt:AMASk:CREate (see page 800)	n/a	n/a
:MTESt:AMASk:SOURce <source> (see page 801)	:MTESt:AMASk:SOURce? (see page 801)	<source> ::= CHANnel<n> <n> ::= {1 2 3 4} for 4ch models <n> ::= {1 2} for 2ch models
:MTESt:AMASk:UNITs <units> (see page 802)	:MTESt:AMASk:UNITs? (see page 802)	<units> ::= {CURRent DIVisions}

Table 26 :MTESt Commands Summary (continued)

Command	Query	Options and Query Returns
:MTESt:AMASk:XDELta <value> (see page 803)	:MTESt:AMASk:XDELta? (see page 803)	<value> ::= X delta value in NR3 format
:MTESt:AMASk:YDELta <value> (see page 804)	:MTESt:AMASk:YDELta? (see page 804)	<value> ::= Y delta value in NR3 format
n/a	:MTESt:COUNT:FWAVEforms? [CHANnel<n>] (see page 805)	<failed> ::= number of failed waveforms in NR1 format
:MTESt:COUNT:RESet (see page 806)	n/a	n/a
n/a	:MTESt:COUNT:TIME? (see page 807)	<time> ::= elapsed seconds in NR3 format
n/a	:MTESt:COUNT:WAVEform s? (see page 808)	<count> ::= number of waveforms in NR1 format
:MTESt:DATA <mask> (see page 809)	:MTESt:DATA? (see page 809)	<mask> ::= data in IEEE 488.2 # format.
:MTESt:DELeTe (see page 810)	n/a	n/a
:MTESt:ENABLe {{0 OFF} {1 ON}} (see page 811)	:MTESt:ENABLe? (see page 811)	{0 1}
:MTESt:LOCK {{0 OFF} {1 ON}} (see page 812)	:MTESt:LOCK? (see page 812)	{0 1}
:MTESt:RMODE <rmode> (see page 813)	:MTESt:RMODE? (see page 813)	<rmode> ::= {FORever TIME SIGMa WAVEforms}
:MTESt:RMODE:FACTION:MEASure {{0 OFF} {1 ON}} (see page 814)	:MTESt:RMODE:FACTION:MEASure? (see page 814)	{0 1}
:MTESt:RMODE:FACTION:PRINT {{0 OFF} {1 ON}} (see page 815)	:MTESt:RMODE:FACTION:PRINT? (see page 815)	{0 1}
:MTESt:RMODE:FACTION:SAVE {{0 OFF} {1 ON}} (see page 816)	:MTESt:RMODE:FACTION:SAVE? (see page 816)	{0 1}
:MTESt:RMODE:FACTION:STOP {{0 OFF} {1 ON}} (see page 817)	:MTESt:RMODE:FACTION:STOP? (see page 817)	{0 1}

Table 26 :MTESt Commands Summary (continued)

Command	Query	Options and Query Returns
:MTESt:RMODe:SIGMa <level> (see page 818)	:MTESt:RMODe:SIGMa? (see page 818)	<level> ::= from 0.1 to 9.3 in NR3 format
:MTESt:RMODe:TIME <seconds> (see page 819)	:MTESt:RMODe:TIME? (see page 819)	<seconds> ::= from 1 to 86400 in NR3 format
:MTESt:RMODe:WAVEform s <count> (see page 820)	:MTESt:RMODe:WAVEform s? (see page 820)	<count> ::= number of waveforms in NR1 format
:MTESt:SCALe:BIND { {0 OFF} {1 ON} } (see page 821)	:MTESt:SCALe:BIND? (see page 821)	{0 1}
:MTESt:SCALe:X1 <x1_value> (see page 822)	:MTESt:SCALe:X1? (see page 822)	<x1_value> ::= X1 value in NR3 format
:MTESt:SCALe:XDELta <xdelta_value> (see page 823)	:MTESt:SCALe:XDELta? (see page 823)	<xdelta_value> ::= X delta value in NR3 format
:MTESt:SCALe:Y1 <y1_value> (see page 824)	:MTESt:SCALe:Y1? (see page 824)	<y1_value> ::= Y1 value in NR3 format
:MTESt:SCALe:Y2 <y2_value> (see page 825)	:MTESt:SCALe:Y2? (see page 825)	<y2_value> ::= Y2 value in NR3 format
:MTESt:SOURce <source> (see page 826)	:MTESt:SOURce? (see page 826)	<source> ::= {CHANnel<n> NONE} <n> ::= {1 2 3 4} for 4ch models <n> ::= {1 2} for 2ch models
n/a	:MTESt:TITLe? (see page 827)	<title> ::= a quoted string of up to 128 ASCII characters

Table 27 :POD<n> Commands Summary

Command	Query	Options and Query Returns
:POD<n>:DISPlay { {0 OFF} {1 ON} } (see page 831)	:POD<n>:DISPlay? (see page 831)	{0 1} <n> ::= 1-2 in NR1 format
:POD<n>:NIBBle<n>:THR eshold <value> (see page 832)	:POD<n>:NIBBle<n>:THR eshold? (see page 832)	<n> ::= 1-2 in NR1 format <value> ::= {CMOS ECL TTL <decimal>[<suffix>]} <decimal> ::= -8.00 to +8.00 in NR3 format [suffix] ::= {V mV uV }
:POD<n>:SIZE <value> (see page 834)	:POD<n>:SIZE? (see page 834)	<value> ::= {SMALl MEDium LARGe}
:POD<n>:THReshold <value> (see page 835)	:POD<n>:THReshold? (see page 835)	<n> ::= 1-2 in NR1 format <value> ::= {CMOS ECL TTL <decimal>[<suffix>]} <decimal> ::= -8.00 to +8.00 in NR3 format [suffix] ::= {V mV uV }

Table 28 :POWer Commands Summary

Command	Query	Options and Query Returns
n/a	:POWer:CLResponse? (see page 845)	n/a
:POWer:CLResponse:APP Ly (see page 846)	n/a	n/a
n/a	:POWer:CLResponse:DATA? [SWEep SINGle] (see page 847)	<binary_block> ::= comma-separated data with newlines at the end of each row
n/a	:POWer:CLResponse:DATA:GMARgin? (see page 848)	<gain_margin> ::= gain margin in dB in NR3 format.
n/a	:POWer:CLResponse:DATA:GMARgin:FREQuency? (see page 849)	<frequency> ::= 0 degrees phase crossover frequency in Hz in NR3 format
n/a	:POWer:CLResponse:DATA:PMARgin? (see page 850)	<phase_margin> ::= phase margin in degrees in NR3 format.

Table 28 :POWER Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:POWER:CLResponse:DATA:PMARgin:FREQuency? (see page 851)	<frequency> ::= 0dB gain crossover frequency in Hz in NR3 format.
:POWER:CLResponse:FREQuency:MODE <mode> (see page 852)	:POWER:CLResponse:FREQuency:MODE? (see page 852)	<mode> ::= {SWEep SINGle}
:POWER:CLResponse:FREQuency:SINGle <value>[suffix] (see page 853)	:POWER:CLResponse:FREQuency:SINGle? (see page 853)	<value> ::= {20 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWER:CLResponse:FREQuency:START <value>[suffix] (see page 854)	:POWER:CLResponse:FREQuency:START? (see page 854)	<value> ::= {20 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:POWER:CLResponse:FREQuency:STOP <value>[suffix] (see page 855)	:POWER:CLResponse:FREQuency:STOP? (see page 855)	<value> ::= {100 1000 10000 100000 1000000 10000000 200000000} [suffix] ::= {Hz kHz MHz}
:POWER:CLResponse:PPDecade <pts> (see page 856)	:POWER:CLResponse:PPDecade? (see page 856)	<pts> ::= {10 20 30 40 50 60 70 80 90 100}
:POWER:CLResponse:SOURce:INPut <source> (see page 857)	:POWER:CLResponse:SOURce:INPut? (see page 857)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:CLResponse:SOURce:OUTPut <source> (see page 858)	:POWER:CLResponse:SOURce:OUTPut? (see page 858)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWER:CLResponse:TRACe <selection> (see page 859)	:POWER:CLResponse:TRACe? (see page 859)	<selection> ::= {NONE ALL GAIN PHASE}[,{GAIN PHASE}]
:POWER:CLResponse:WGEN:LOAD <impedance> (see page 860)	:POWER:CLResponse:WGEN:LOAD? (see page 860)	<impedance> ::= {ONEMeg FIFTy}
:POWER:CLResponse:WGEN:VOLTage <amplitude>[,<range>] (see page 861)	:POWER:CLResponse:WGEN:VOLTage? [<range>] (see page 861)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F10MILLIHZ F100MILLIHZ F1HZ {F10HZ F20HZ} F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ {F20MHZ F100MHZ}}

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:CLResponse:WGEN:VOLTage:PROFile {{0 OFF} {1 ON}} (see page 862)	:POWer:CLResponse:WGEN:VOLTage:PROFile? (see page 862)	{0 1}
:POWer:DESKew (see page 863)	n/a	n/a
:POWer:EFFiciency:APPLY (see page 864)	n/a	n/a
:POWer:EFFiciency:TYPE <type> (see page 865)	:POWer:EFFiciency:TYPE? (see page 865)	<type> ::= {DCDC DCAC ACDC ACAC}
:POWer:ENABLE {{0 OFF} {1 ON}} (see page 866)	:POWer:ENABLE? (see page 866)	{0 1}
:POWer:HARMonics:APPLY (see page 867)	n/a	n/a
n/a	:POWer:HARMonics:DATA? (see page 868)	<binary_block> ::= comma-separated data with newlines at the end of each row
:POWer:HARMonics:DISPLAY <display> (see page 869)	:POWer:HARMonics:DISPLAY? (see page 869)	<display> ::= {TABLE BAR OFF}
n/a	:POWer:HARMonics:FAILcount? (see page 870)	<count> ::= integer in NR1 format
:POWer:HARMonics:LINE <frequency> (see page 871)	:POWer:HARMonics:LINE? (see page 871)	<frequency> ::= {F50 F60 F400 AUTO}
n/a	:POWer:HARMonics:POWERfactor? (see page 872)	<value> ::= Class C power factor in NR3 format
:POWer:HARMonics:RPOWER <source> (see page 873)	:POWer:HARMonics:RPOWER? (see page 873)	<source> ::= {MEASured USER}
:POWer:HARMonics:RPOWER:USER <value> (see page 874)	:POWer:HARMonics:RPOWER:USER? (see page 874)	<value> ::= Watts from 1.0 to 600.0 in NR3 format
n/a	:POWer:HARMonics:RUNcount? (see page 875)	<count> ::= integer in NR1 format

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:HARMonics:STANdard <class> (see page 876)	:POWer:HARMonics:STANdard? (see page 876)	<class> ::= {A B C D}
n/a	:POWer:HARMonics:STATus? (see page 877)	<status> ::= {PASS FAIL UNTested}
n/a	:POWer:HARMonics:THD? (see page 878)	<value> ::= Total Harmonics Distortion in NR3 format
:POWer:INRush:APPLy (see page 879)	n/a	n/a
:POWer:INRush:EXIT (see page 880)	n/a	n/a
:POWer:INRush:NEXT (see page 881)	n/a	n/a
:POWer:ITYPE <type> (see page 882)	:POWer:ITYPE? (see page 882)	<type> ::= {DC AC}
:POWer:MODulation:APPLy (see page 883)	n/a	n/a
:POWer:MODulation:SOURce <source> (see page 884)	:POWer:MODulation:SOURce? (see page 884)	<source> ::= {V I}
:POWer:MODulation:TYPE <modulation> (see page 885)	:POWer:MODulation:TYPE? (see page 885)	<modulation> ::= {VAverage ACRMs VRATio PERiod FREQuency PWIDith NWIDth DUTYcycle NDUTYcycle RISetime FALLtime}
:POWer:ONOff:APPLy (see page 887)	n/a	n/a
:POWer:ONOff:EXIT (see page 888)	n/a	n/a
:POWer:ONOff:NEXT (see page 889)	n/a	n/a
:POWer:ONOff:TEST {{0 OFF} {1 ON}} (see page 890)	:POWer:ONOff:TEST? (see page 890)	{0 1}
:POWer:ONOff:THReshol ds <type>, <input_thr>, <output_thr> (see page 891)	:POWer:ONOff:THReshol ds? <type> (see page 891)	<type> ::= {0 1} <input_thr> ::= percent from 0-100 in NR1 format <output_thr> ::= percent from 0-100 in NR1 format

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:POWer:PSRR? (see page 893)	n/a
:POWer:PSRR:APPLy (see page 894)	n/a	n/a
n/a	:POWer:PSRR:DATA? [SWEep SINGle] (see page 895)	<binary_block> ::= comma-separated data with newlines at the end of each row
:POWer:PSRR:FREQuency:MAXimum <value>[suffix] (see page 896)	:POWer:PSRR:FREQuency:MAXimum? (see page 896)	<value> ::= {10 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:FREQuency:MINimum <value>[suffix] (see page 897)	:POWer:PSRR:FREQuency:MINimum? (see page 897)	<value> ::= {1 10 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:FREQuency:MODE <mode> (see page 898)	:POWer:PSRR:FREQuency:MODE? (see page 898)	<mode> ::= {SWEep SINGle}
:POWer:PSRR:FREQuency:SINGle <value>[suffix] (see page 899)	:POWer:PSRR:FREQuency:SINGle? (see page 899)	<value> ::= {1 10 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:PPDecade <pts> (see page 900)	:POWer:PSRR:PPDecade? (see page 900)	<pts> ::= {10 20 30 40 50 60 70 80 90 100}
:POWer:PSRR:SOURce:INPut <source> (see page 901)	:POWer:PSRR:SOURce:INPut? (see page 901)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:PSRR:SOURce:OUTPut <source> (see page 902)	:POWer:PSRR:SOURce:OUTPut? (see page 902)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:PSRR:TRACe <selection> (see page 903)	:POWer:PSRR:TRACe? (see page 903)	<selection> ::= {NONE GAIN}
:POWer:PSRR:WGEN:LOAD <impedance> (see page 904)	:POWer:PSRR:WGEN:LOAD? (see page 904)	<impedance> ::= {ONEMeg FIFTy}

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:PSRR:WGEN:VOLT age <amplitude>[,<range>] (see page 905)	:POWer:PSRR:WGEN:VOLT age? [<range>] (see page 905)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F10MILLIHZ F100MILLIHZ F1HZ {F10HZ F20HZ} F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ {F20MHZ F100MHZ}}
:POWer:PSRR:WGEN:VOLT age:PROFile {{0 OFF} {1 ON}} (see page 906)	:POWer:PSRR:WGEN:VOLT age:PROFile? (see page 906)	{0 1}
:POWer:QUALity:APPLy (see page 907)	n/a	n/a
:POWer:RIPple:APPLy (see page 908)	n/a	n/a
:POWer:SIGNals:AUTose tup <analysis> (see page 909)	n/a	<analysis> ::= {HARMonics EFFiciency RIPple MODulation QUALity SLEW SWITCh RDSVce}
:POWer:SIGNals:CYCLes :HARMonics <count> (see page 910)	:POWer:SIGNals:CYCLes :HARMonics? (see page 910)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWer:SIGNals:CYCLes :QUALity <count> (see page 911)	:POWer:SIGNals:CYCLes :QUALity? (see page 911)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWer:SIGNals:DURati on:EFFiciency <value>[suffix] (see page 912)	:POWer:SIGNals:DURati on:EFFiciency? (see page 912)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNals:DURati on:MODulation <value>[suffix] (see page 913)	:POWer:SIGNals:DURati on:MODulation? (see page 913)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNals:DURati on:ONOff:OFF <value>[suffix] (see page 914)	:POWer:SIGNals:DURati on:ONOff:OFF? (see page 914)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNals:DURati on:ONOff:ON <value>[suffix] (see page 915)	:POWer:SIGNals:DURati on:ONOff:ON? (see page 915)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}

Table 28 :POWER Commands Summary (continued)

Command	Query	Options and Query Returns
:POWER:SIGNals:DURati on:RIPPlE <value>[suffix] (see page 916)	:POWER:SIGNals:DURati on:RIPPlE? (see page 916)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWER:SIGNals:DURati on:TRANSient <value>[suffix] (see page 917)	:POWER:SIGNals:DURati on:TRANSient? (see page 917)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWER:SIGNals:IEXPe cted <value>[suffix] (see page 918)	:POWER:SIGNals:IEXPe cted? (see page 918)	<value> ::= Expected current value in NR3 format [suffix] ::= {A mA}
:POWER:SIGNals:OVERsh oot <percent> (see page 919)	:POWER:SIGNals:OVERsh oot? (see page 919)	<percent> ::= percent of overshoot value in NR1 format [suffix] ::= {V mV}
:POWER:SIGNals:VMAXim um:INRush <value>[suffix] (see page 920)	:POWER:SIGNals:VMAXim um:INRush? (see page 920)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWER:SIGNals:VMAXim um:ONOff:OFF <value>[suffix] (see page 921)	:POWER:SIGNals:VMAXim um:ONOff:OFF? (see page 921)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWER:SIGNals:VMAXim um:ONOff:ON <value>[suffix] (see page 922)	:POWER:SIGNals:VMAXim um:ONOff:ON? (see page 922)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWER:SIGNals:VSTead y:ONOff:OFF <value>[suffix] (see page 923)	:POWER:SIGNals:VSTead y:ONOff:OFF? (see page 923)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}
:POWER:SIGNals:VSTead y:ONOff:ON <value>[suffix] (see page 924)	:POWER:SIGNals:VSTead y:ONOff:ON? (see page 924)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}
:POWER:SIGNals:VSTead y:TRANSient <value>[suffix] (see page 925)	:POWER:SIGNals:VSTead y:TRANSient? (see page 925)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:SIGNals:SOURce :CURRent<i> <source> (see page 926)	:POWer:SIGNals:SOURce :CURRent<i>? (see page 926)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:SIGNals:SOURce :VOLTag<i> <source> (see page 927)	:POWer:SIGNals:SOURce :VOLTag<i>? (see page 927)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:SLEW:APPLy (see page 928)	n/a	n/a
:POWer:SLEW:SOURce <source> (see page 929)	:POWer:SLEW:SOURce? (see page 929)	<source> ::= {V I}
:POWer:SWITCh:APPLy (see page 930)	n/a	n/a
:POWer:SWITCh:CONduct ion <conduction> (see page 931)	:POWer:SWITCh:CONduct ion? (see page 931)	<conduction> ::= {WAVEform RDS VCE}
:POWer:SWITCh:IREFere nce <percent> (see page 932)	:POWer:SWITCh:IREFere nce? (see page 932)	<percent> ::= percent in NR1 format
:POWer:SWITCh:RDS <value>[suffix] (see page 933)	:POWer:SWITCh:RDS? (see page 933)	<value> ::= Rds(on) value in NR3 format [suffix] ::= {OHM mOHM}
:POWer:SWITCh:VCE <value>[suffix] (see page 934)	:POWer:SWITCh:VCE? (see page 934)	<value> ::= Vce(sat) value in NR3 format [suffix] ::= {V mV}
:POWer:SWITCh:VREFere nce <percent> (see page 935)	:POWer:SWITCh:VREFere nce? (see page 935)	<percent> ::= percent in NR1 format
:POWer:TRANSient:APPL y (see page 936)	n/a	n/a
:POWer:TRANSient:EXIT (see page 937)	n/a	n/a
:POWer:TRANSient:IINi tial <value>[suffix] (see page 938)	:POWer:TRANSient:IINi tial? (see page 938)	<value> ::= Initial current value in NR3 format [suffix] ::= {A mA}

Table 28 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:TRANsient:INew <value>[suffix] (see page 939)	:POWer:TRANsient:INew ? (see page 939)	<value> ::= New current value in NR3 format [suffix] ::= {A mA}
:POWer:TRANsient:NEXT (see page 940)	n/a	n/a

Table 29 :RECall Commands Summary

Command	Query	Options and Query Returns
:RECall:ARBitrary:[STArT] [<internal_fname> <external_fname>]][, <column>][, <wavegen_id>] (see page 944)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected <column> ::= Column in CSV file to load. Column number starts from 1. <wavegen_id> ::= WGEN1
:RECall:DBc[:STArT] [<file_name>] [, <protocol_decode_bus>] (see page 945)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".dbc". <protocol_decode_bus> ::= {PBUS<n> SBUS<n>} <n> ::= 1 to (# of protocol decode bus) in NR1 format
:RECall:FIleName <base_name> (see page 946)	:RECall:FIleName? (see page 946)	<base_name> ::= quoted ASCII string

Table 29 :RECall Commands Summary (continued)

Command	Query	Options and Query Returns
:RECall:LDF[:START] [<file_name>] [, <protocol_decode_bus>] (see page 947)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".ldf". <protocol_decode_bus> ::= {PBUS<n> SBUS<n>} <n> ::= 1 to (# of protocol decode bus) in NR1 format
:RECall:MASK[:START] [<internal_fname> <external_fname>]] (see page 948)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:RECall:PWD <path_name> (see page 949)	:RECall:PWD? (see page 949)	<path_name> ::= quoted ASCII string
:RECall:SETup[:START] [<internal_fname> <external_fname>]] (see page 950)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:RECall:WMEemory<r>[:S TART] [<file_name> <data>] (see page 951)	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". <data> ::= binary block data in IEEE 488.2 # format

Table 30 :SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:ARBitrary:[STARt] [{<internal_fname> <external_fname>}] [, <wavegen_id>] (see page 957)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected <wavegen_id> ::= WGEN1
:SAVE:FILEname <base_name> (see page 958)	:SAVE:FILEname? (see page 958)	<base_name> ::= quoted ASCII string
:SAVE:IMAGe[:STARt] [<file_name>] (see page 959)	n/a	<file_name> ::= quoted ASCII string
:SAVE:IMAGe:FACTors {{0 OFF} {1 ON}} (see page 960)	:SAVE:IMAGe:FACTors? (see page 960)	{0 1}
:SAVE:IMAGe:FORMat <format> (see page 961)	:SAVE:IMAGe:FORMat? (see page 961)	<format> ::= {{BMP BMP24bit} PNG NONE}
:SAVE:LISTer[:STARt] [<file_name>] (see page 962)	n/a	<file_name> ::= quoted ASCII string
:SAVE:MASK[:STARt] [{<internal_fname> <external_fname>}] (see page 963)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:SAVE:MULTi[:STARt] [<file_name>] (see page 964)	n/a	<file_name> ::= quoted ASCII string
:SAVE:POWer[:STARt] [<file_spec>] (see page 965)	n/a	<file_name> ::= quoted ASCII string

Table 30 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:PWD <path_name> (see page 966)	:SAVE:PWD? (see page 966)	<path_name> ::= quoted ASCII string
:SAVE:RESults:[START] [<file_spec>] (see page 967)	n/a	<file_name> ::= quoted ASCII string
:SAVE:RESults:FORMat: CURSor {{0 OFF} {1 ON}} (see page 968)	:SAVE:RESults:FORMat: CURSor? (see page 968)	{0 1}
:SAVE:RESults:FORMat: HISTogram {{0 OFF} {1 ON}} (see page 969)	:SAVE:RESults:FORMat: HISTogram? (see page 969)	{0 1}
:SAVE:RESults:FORMat: MASK {{0 OFF} {1 ON}} (see page 970)	:SAVE:RESults:FORMat: MASK? (see page 970)	{0 1}
:SAVE:RESults:FORMat: MEASurement {{0 OFF} {1 ON}} (see page 971)	:SAVE:RESults:FORMat: MEASurement? (see page 971)	{0 1}
:SAVE:RESults:FORMat: SEARch {{0 OFF} {1 ON}} (see page 972)	:SAVE:RESults:FORMat: SEARch? (see page 972)	{0 1}
:SAVE:RESults:FORMat: SEGmented {{0 OFF} {1 ON}} (see page 973)	:SAVE:RESults:FORMat: SEGmented? (see page 973)	{0 1}
:SAVE:SETup[:START] [<internal_fname> <external_fname>]] (see page 974)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:SAVE:WAVEform[:START] [<file_name>] (see page 975)	n/a	<file_name> ::= quoted ASCII string

Table 30 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:WAVEform:FORMat <format> (see page 976)	:SAVE:WAVEform:FORMat ? (see page 976)	<format> ::= {ASCIixy CSV BINary NONE}
:SAVE:WAVEform:LENGth <length> (see page 977)	:SAVE:WAVEform:LENGth ? (see page 977)	<length> ::= 100 to max. length; an integer in NR1 format
:SAVE:WAVEform:LENGth :MAX {{0 OFF} {1 ON}} (see page 978)	:SAVE:WAVEform:LENGth :MAX? (see page 978)	{0 1}
:SAVE:WAVEform:SEGMen ted <option> (see page 979)	:SAVE:WAVEform:SEGMen ted? (see page 979)	<option> ::= {ALL CURRent}
:SAVE:WMEMemory:SOURce <source> (see page 980)	:SAVE:WMEMemory:SOURce? (see page 980)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMemory<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:WMEMemory[:START] [<file_name>] (see page 981)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".h5".

Table 31 General :SBUS<n> Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:DISPlay {{0 OFF} {1 ON}} (see page 986)	:SBUS<n>:DISPlay? (see page 986)	{0 1}
:SBUS<n>:MODE <mode> (see page 987)	:SBUS<n>:MODE? (see page 987)	<mode> ::= {A429 CAN IIC LIN M1553 SENT SPI UART USB USBPd}

Table 32 :SBUS<n>:A429 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:A429:AUTosetup (see page 990)	n/a	n/a
:SBUS<n>:A429:BASE<base> (see page 991)	:SBUS<n>:A429:BASE? (see page 991)	<base> ::= {BINary HEX}
:SBUS<n>:A429:BAUDrate <baudrate> (see page 992)	:SBUS<n>:A429:BAUDrate? (see page 992)	<baudrate> ::= integer from 10000 to 1000000
:SBUS<n>:A429:FORMat<format> (see page 993)	:SBUS<n>:A429:FORMat? (see page 993)	<format> ::= {LDSDi LDSSm LDATa}
:SBUS<n>:A429:SIGNal<signal> (see page 994)	:SBUS<n>:A429:SIGNal? (see page 994)	<signal> ::= {A B DIFFerential}
:SBUS<n>:A429:SOURce<source> (see page 995)	:SBUS<n>:A429:SOURce? (see page 995)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:A429:SPEed<speed> (see page 996)	:SBUS<n>:A429:SPEed? (see page 996)	<speed> ::= {LOW HIGH USER}
:SBUS<n>:A429:TRIGger:LABel <value> (see page 997)	:SBUS<n>:A429:TRIGger:LABel? (see page 997)	<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 998)	:SBUS<n>:A429:TRIGger:PATtern:DATA? (see page 998)	<string> ::= "nn...n" where n ::= {0 1 X}, length depends on FORMat
:SBUS<n>:A429:TRIGger:PATtern:SDI<string> (see page 999)	:SBUS<n>:A429:TRIGger:PATtern:SDI? (see page 999)	<string> ::= "nn" where n ::= {0 1 X}, length always 2 bits
:SBUS<n>:A429:TRIGger:PATtern:SSM<string> (see page 1000)	:SBUS<n>:A429:TRIGger:PATtern:SSM? (see page 1000)	<string> ::= "nn" where n ::= {0 1 X}, length always 2 bits

Table 32 :SBUS<n>:A429 Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:A429:TRIGger :RANGe <min>,<max> (see page 1001)	:SBUS<n>:A429:TRIGger :RANGe? (see page 1001)	<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 1002)	:SBUS<n>:A429:TRIGger :TYPE? (see page 1002)	<condition> ::= {WSTArt WSTOp LABEL LBITs PERRor WERRor GERRor WGERrors ALLerrors LRANGe ABITs AOBits AZBits}

Table 33 :SBUS<n>:CAN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:CAN:DISPlay <type> (see page 1007)	:SBUS<n>:CAN:DISPlay? (see page 1007)	<type> ::= {HEXadecimal SYMBolic}
:SBUS<n>:CAN:FDSPoint <value> (see page 1008)	:SBUS<n>:CAN:FDSPoint ? (see page 1008)	<value> ::= even numbered percentages from 30 to 90 in NR3 format.
:SBUS<n>:CAN:SAMPlEpo int <percent> (see page 1009)	:SBUS<n>:CAN:SAMPlEpo int? (see page 1009)	<percent> ::= 30.0 to 90.0 in NR3 format
:SBUS<n>:CAN:SIGNAL:B AUDrate <baudrate> (see page 1010)	:SBUS<n>:CAN:SIGNAL:B AUDrate? (see page 1010)	<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments, or 5000000
:SBUS<n>:CAN:SIGNAL:D EFinition <value> (see page 1011)	:SBUS<n>:CAN:SIGNAL:D EFinition? (see page 1011)	<value> ::= {CANH CANL RX TX DIFFerential DIPL DIFH}
:SBUS<n>:CAN:SIGNAL:F DBaudrate <baudrate> (see page 1012)	:SBUS<n>:CAN:SIGNAL:F DBaudrate? (see page 1012)	<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.
:SBUS<n>:CAN:SIGNAL:X LBaudrate <baudrate> (see page 1013)	:SBUS<n>:CAN:SIGNAL:X LBaudrate? (see page 1013)	<baudrate> ::= integer from 10000 to 20000000 in 100 b/s increments.

Table 33 :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:SOURce <source> (see page 1014)	:SBUS<n>:CAN:SOURce? (see page 1014)	<source> ::= {CHANnel<n> DIGital<d> } <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:CAN:TRIGger <condition> (see page 1015)	:SBUS<n>:CAN:TRIGger? (see page 1016)	<condition> ::= {SOF EOF IDData DATA IDRemote IDEither ERRor ACKerror FORMerror STUFferor CRCerror SPECerror ALLerrors BRsBit CRCDbit EBActive EBPAssive OVERload MESSage MSIGnal}
:SBUS<n>:CAN:TRIGger:IDFilter {{0 OFF} {1 ON}} (see page 1018)	:SBUS<n>:CAN:TRIGger:IDFilter? (see page 1018)	{0 1}
:SBUS<n>:CAN:TRIGger:PATtern:DATA <string> (see page 1019)	:SBUS<n>:CAN:TRIGger:PATtern:DATA? (see page 1019)	<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 <dcl> (see page 1020)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:DLC? (see page 1020)	<dcl> ::= integer between -1 (don't care) and 64, in NR1 format.
:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth <length> (see page 1021)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth? (see page 1021)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:CAN:TRIGger:PATtern:DATA:START <start> (see page 1022)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:START? (see page 1022)	<start> ::= integer between 0 and 63, in NR1 format.
:SBUS<n>:CAN:TRIGger:PATtern:ID <string> (see page 1023)	:SBUS<n>:CAN:TRIGger:PATtern:ID? (see page 1023)	<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 1024)	:SBUS<n>:CAN:TRIGger:PATtern:ID:MODE? (see page 1024)	<value> ::= {STANDard EXTended}

Table 33 :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge<name> (see page 1025)	:SBUS<n>:CAN:TRIGger:SYMBOLic:MESSAge? (see page 1025)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger:SYMBOLic:SIGNAL<name> (see page 1026)	:SBUS<n>:CAN:TRIGger:SYMBOLic:SIGNAL? (see page 1026)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue<data> (see page 1027)	:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue? (see page 1027)	<data> ::= value in NR3 format
:SBUS<n>:CAN:TRIGger:TYPE<type> (see page 1028)	:SBUS<n>:CAN:TRIGger:TYPE? (see page 1028)	<type> ::= {STANDARD FD XL}
:SBUS<n>:CAN:TYPE<type> (see page 1029)	:SBUS<n>:CAN:TYPE? (see page 1029)	<type> ::= {STANDARD FD XFAST XSIC}
:SBUS<n>:CAN:XLSPoint<value> (see page 1030)	:SBUS<n>:CAN:XLSPoint? (see page 1030)	<value> ::= even numbered percentages from 30 to 90 in NR3 format.

Table 34 :SBUS<n>:IIC Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:IIC:ASIZE<size> (see page 1032)	:SBUS<n>:IIC:ASIZE? (see page 1032)	<size> ::= {BIT7 BIT8}
:SBUS<n>:IIC[:SOURce]:CLOCK<source> (see page 1033)	:SBUS<n>:IIC[:SOURce]:CLOCK? (see page 1033)	<source> ::= {CHANnel<n> DIGital<d> } <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 1034)	:SBUS<n>:IIC[:SOURce]:DATA? (see page 1034)	<source> ::= {CHANnel<n> DIGital<d> } <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 1035)	:SBUS<n>:IIC:TRIGger:PATTERN:ADDRESS? (see page 1035)	<value> ::= integer

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

Command	Query	Options and Query Returns
:SBUS<n>:IIC:TRIGger:PATtern:DATA <value> (see page 1036)	:SBUS<n>:IIC:TRIGger:PATtern:DATA? (see page 1036)	<value> ::= integer
:SBUS<n>:IIC:TRIGger:PATtern:DATA2 <value> (see page 1037)	:SBUS<n>:IIC:TRIGger:PATtern:DATA2? (see page 1037)	<value> ::= integer
:SBUS<n>:IIC:TRIGger:QUALifier <value> (see page 1038)	:SBUS<n>:IIC:TRIGger:QUALifier? (see page 1038)	<value> ::= {EQUAL NOTequal LESSthan GREATERthan}
:SBUS<n>:IIC:TRIGger[:TYPE] <type> (see page 1039)	:SBUS<n>:IIC:TRIGger[:TYPE]? (see page 1039)	<type> ::= {START STOP REStart ADDRESS ANACK DNACK NACKnowledge REPRom READ7 WRITe7 R7Data2 W7Data2 WRITe10}

Table 35 :SBUS<n>:LIN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:LIN:DISPlay <type> (see page 1043)	:SBUS<n>:LIN:DISPlay? (see page 1043)	<type> ::= {HEXadecimal SYMBolic}
:SBUS<n>:LIN:PARity {{0 OFF} {1 ON}} (see page 1044)	:SBUS<n>:LIN:PARity? (see page 1044)	{0 1}
:SBUS<n>:LIN:SAMplepo int <value> (see page 1045)	:SBUS<n>:LIN:SAMplepo int? (see page 1045)	<value> ::= {60 62.5 68 70 75 80 87.5} in NR3 format
:SBUS<n>:LIN:SIGNAL:B AUDrate <baudrate> (see page 1046)	:SBUS<n>:LIN:SIGNAL:B AUDrate? (see page 1046)	<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments
:SBUS<n>:LIN:SOURce <source> (see page 1047)	:SBUS<n>:LIN:SOURce? (see page 1047)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:LIN:STANDard <std> (see page 1048)	:SBUS<n>:LIN:STANDard ? (see page 1048)	<std> ::= {LIN13 LIN13NLC LIN20}
:SBUS<n>:LIN:SYNCbrea k <value> (see page 1049)	:SBUS<n>:LIN:SYNCbrea k? (see page 1049)	<value> ::= integer = {11 12 13}

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

Command	Query	Options and Query Returns
:SBUS<n>:LIN:TRIGger<condition> (see page 1050)	:SBUS<n>:LIN:TRIGger? (see page 1050)	<condition> ::= {SYNCbreak ID DATA}
:SBUS<n>:LIN:TRIGger:ID <value> (see page 1051)	:SBUS<n>:LIN:TRIGger:ID? (see page 1051)	<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 1052)	:SBUS<n>:LIN:TRIGger:PATtern:DATA? (see page 1052)	<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 1054)	:SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth? (see page 1054)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:LIN:TRIGger:PATtern:FORMat <base> (see page 1055)	:SBUS<n>:LIN:TRIGger:PATtern:FORMat? (see page 1055)	<base> ::= {BINary HEX DECimal}
:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe <name> (see page 1056)	:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe? (see page 1056)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal <name> (see page 1057)	:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal? (see page 1057)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger:SYMBolic:VALue <data> (see page 1058)	:SBUS<n>:LIN:TRIGger:SYMBolic:VALue? (see page 1058)	<data> ::= value in NR3 format

Table 36 :SBUS<n>:M1553 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:M1553:AUTOsetup (see page 1060)	n/a	n/a
:SBUS<n>:M1553:BASE<base> (see page 1061)	:SBUS<n>:M1553:BASE? (see page 1061)	<base> ::= {BINary HEX}
:SBUS<n>:M1553:SOURce<source> (see page 1062)	:SBUS<n>:M1553:SOURce? (see page 1062)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:M1553:TRIGger:PATtern:DATA<string> (see page 1063)	:SBUS<n>:M1553:TRIGger:PATtern:DATA? (see page 1063)	<string> ::= "nn...n" where n ::= {0 1 X}
:SBUS<n>:M1553:TRIGger:RTA<value> (see page 1064)	:SBUS<n>:M1553:TRIGger:RTA? (see page 1064)	<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:TRIGger:TYPE<type> (see page 1065)	:SBUS<n>:M1553:TRIGger:TYPE? (see page 1065)	<type> ::= {DStart DStOp CStart CStOp RTA PERRor SERRor MERRor RTA11}

Table 37 :SBUS<n>:SENT Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SENT:CLOCK<period> (see page 1069)	:SBUS<n>:SENT:CLOCK? (see page 1069)	<period> ::= the nominal clock period (tick), from 500 ns to 300 us, in NR3 format.
:SBUS<n>:SENT:CRC<format> (see page 1070)	:SBUS<n>:SENT:CRC? (see page 1070)	<format> ::= {LEGacy RECommended}
:SBUS<n>:SENT:DISPlay<base> (see page 1071)	:SBUS<n>:SENT:DISPlay? (see page 1071)	<base> ::= {HEX DECimal SYMBolic}
:SBUS<n>:SENT:FORMat<decode> (see page 1073)	:SBUS<n>:SENT:FORMat? (see page 1073)	<decode> ::= {NIBBles FSIGnal FSSerial FESerial SSERial ESERial}
:SBUS<n>:SENT:IDLE<state> (see page 1075)	:SBUS<n>:SENT:IDLE? (see page 1075)	<state> ::= {LOW HIGH}

Table 37 :SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:LENGth<#_nibbles> (see page 1076)	:SBUS<n>:SENT:LENGth? (see page 1076)	<#_nibbles> ::= from 1-6, in NR1 format.
:SBUS<n>:SENT:PPULse {{0 OFF} {1 ON} SPC} (see page 1077)	:SBUS<n>:SENT:PPULse? (see page 1077)	{0 1 SPC}
:SBUS<n>:SENT:SIGNal<s>:DISPlay {{0 OFF} {1 ON}} (see page 1079)	:SBUS<n>:SENT:SIGNal<s>:DISPlay? (see page 1079)	<s> ::= 1-6, in NR1 format. {0 1}
:SBUS<n>:SENT:SIGNal<s>:LENGth <length> (see page 1080)	:SBUS<n>:SENT:SIGNal<s>:LENGth? (see page 1080)	<s> ::= 1-6, in NR1 format. <length> ::= from 1-24, in NR1 format.
:SBUS<n>:SENT:SIGNal<s>:MULTiplier <multiplier> (see page 1082)	:SBUS<n>:SENT:SIGNal<s>:MULTiplier? (see page 1082)	<s> ::= 1-6, in NR1 format. <multiplier> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNal<s>:OFFSet <offset> (see page 1084)	:SBUS<n>:SENT:SIGNal<s>:OFFSet? (see page 1084)	<s> ::= 1-6, in NR1 format. <offset> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNal<s>:ORDer <order> (see page 1086)	:SBUS<n>:SENT:SIGNal<s>:ORDer? (see page 1086)	<s> ::= 1-6, in NR1 format. <order> ::= {MSNFirst LSNFirst}
:SBUS<n>:SENT:SIGNal<s>:STARt <position> (see page 1088)	:SBUS<n>:SENT:SIGNal<s>:STARt? (see page 1088)	<s> ::= 1-6, in NR1 format. <position> ::= from 0-23, in NR1 format.
:SBUS<n>:SENT:SOURce <source> (see page 1090)	:SBUS<n>:SENT:SOURce? (see page 1090)	<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 1092)	:SBUS<n>:SENT:TOLeran ce? (see page 1092)	<percent> ::= from 3-30, in NR1 format.
:SBUS<n>:SENT:TRIGger <mode> (see page 1093)	:SBUS<n>:SENT:TRIGger ? (see page 1093)	<mode> ::= {SFCMessage SSCMessage FCData SCMid SCData FCCerror SCCerror CRCerror TOLerror PPERror SSPerror}

Table 37 :SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:TRIGger :FAST:DATA <string> (see page 1095)	:SBUS<n>:SENT:TRIGger :FAST:DATA? (see page 1095)	<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 1096)	:SBUS<n>:SENT:TRIGger :SLOW:DATA? (see page 1096)	<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 1098)	:SBUS<n>:SENT:TRIGger :SLOW:ID? (see page 1098)	<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:ILENgth <length> (see page 1100)	:SBUS<n>:SENT:TRIGger :SLOW:ILENgth? (see page 1100)	<length> ::= {SHORT LONG}
:SBUS<n>:SENT:TRIGger :TOLerance <percent> (see page 1101)	:SBUS<n>:SENT:TRIGger :TOLerance? (see page 1101)	<percent> ::= from 1-18, in NR1 format.

Table 38 :SBUS<n>:SPI Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SPI:BITOrder <order> (see page 1104)	:SBUS<n>:SPI:BITOrder ? (see page 1104)	<order> ::= {LSBFirst MSBFirst}
:SBUS<n>:SPI:CLOCK:SL OPe <slope> (see page 1105)	:SBUS<n>:SPI:CLOCK:SL OPe? (see page 1105)	<slope> ::= {NEGative POSitive}
:SBUS<n>:SPI:CLOCK:TI Meout <time_value> (see page 1106)	:SBUS<n>:SPI:CLOCK:TI Meout? (see page 1106)	<time_value> ::= time in seconds in NR3 format
:SBUS<n>:SPI:DElay <value> (see page 1107)	:SBUS<n>:SPI:DElay? (see page 1107)	<value> ::= {OFF 2-63}

Table 38 :SBUS<n>:SPI Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SPI:FRAMing <value> (see page 1108)	:SBUS<n>:SPI:FRAMing? (see page 1108)	<value> ::= {CHIPselect {NCHipselect NOTC} TIMEout}
:SBUS<n>:SPI:SOURce:C LOCK <source> (see page 1109)	:SBUS<n>:SPI:SOURce:C LOCK? (see page 1109)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:F RAME <source> (see page 1110)	:SBUS<n>:SPI:SOURce:F RAME? (see page 1110)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:M ISO <source> (see page 1111)	:SBUS<n>:SPI:SOURce:M ISO? (see page 1111)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:M OSI <source> (see page 1112)	:SBUS<n>:SPI:SOURce:M OSI? (see page 1112)	<value> ::= {CHANnel<n> DIGital<d>} <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 page 1113)	:SBUS<n>:SPI:TRIGger: PATtern:MISO:DATA? (see page 1113)	<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 page 1114)	:SBUS<n>:SPI:TRIGger: PATtern:MISO:WIDTh? (see page 1114)	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger: PATtern:MOSI:DATA <string> (see page 1115)	:SBUS<n>:SPI:TRIGger: PATtern:MOSI:DATA? (see page 1115)	<string> ::= "nn...n" where n ::= {0 1 X \$} <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X \$}

Table 38 :SBUS<n>:SPI Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh<width> (see page 1116)	:SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh? (see page 1116)	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger:TYPE <value> (see page 1117)	:SBUS<n>:SPI:TRIGger:TYPE? (see page 1117)	<value> ::= {MOSI MISO}
:SBUS<n>:SPI:WIDTh<word_width> (see page 1118)	:SBUS<n>:SPI:WIDTh? (see page 1118)	<word_width> ::= integer 4-16 in NR1 format

Table 39 :SBUS<n>:UART Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:UART:BASE<base> (see page 1121)	:SBUS<n>:UART:BASE? (see page 1121)	<base> ::= {ASCIi BINary HEX}
:SBUS<n>:UART:BAUDrate <baudrate> (see page 1122)	:SBUS<n>:UART:BAUDrate? (see page 1122)	<baudrate> ::= integer from 100 to 8000000, 10000000, or 12000000
:SBUS<n>:UART:BITorder <bitorder> (see page 1123)	:SBUS<n>:UART:BITorder? (see page 1123)	<bitorder> ::= {LSBFirst MSBFirst}
:SBUS<n>:UART:FRAMing<value> (see page 1124)	:SBUS<n>:UART:FRAMing? (see page 1124)	<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 1125)	:SBUS<n>:UART:PARity? (see page 1125)	<parity> ::= {EVEN ODD NONE}
:SBUS<n>:UART:POLarity <polarity> (see page 1126)	:SBUS<n>:UART:POLarity? (see page 1126)	<polarity> ::= {HIGH LOW}

Table 39 :SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:UART:SOURce:RX <source> (see page 1127)	:SBUS<n>:UART:SOURce:RX? (see page 1127)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:UART:SOURce:TX <source> (see page 1128)	:SBUS<n>:UART:SOURce:TX? (see page 1128)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:UART:TRIGger:BASE <base> (see page 1129)	:SBUS<n>:UART:TRIGger:BASE? (see page 1129)	<base> ::= {ASCii HEX}
:SBUS<n>:UART:TRIGger:BURSt <value> (see page 1130)	:SBUS<n>:UART:TRIGger:BURSt? (see page 1130)	<value> ::= {OFF 1 to 4096 in NR1 format}
:SBUS<n>:UART:TRIGger:DATA <value> (see page 1131)	:SBUS<n>:UART:TRIGger:DATA? (see page 1131)	<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 1132)	:SBUS<n>:UART:TRIGger:IDLE? (see page 1132)	<time_value> ::= time from 1 us to 10 s in NR3 format
:SBUS<n>:UART:TRIGger:QUALifier <value> (see page 1133)	:SBUS<n>:UART:TRIGger:QUALifier? (see page 1133)	<value> ::= {EQUal NOTequal GREaterthan LESSthan}

Table 39 :SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:UART:TRIGger :TYPE <value> (see page 1134)	:SBUS<n>:UART:TRIGger :TYPE? (see page 1134)	<value> ::= {RSTart RSTop RDATa RD1 RD0 RDX RXParity TXParity TSTart TSTop TDATa TD1 TD0 TDX}
:SBUS<n>:UART:WIDTh <width> (see page 1135)	:SBUS<n>:UART:WIDTh? (see page 1135)	<width> ::= {5 6 7 8 9}

Table 40 :SBUS<n>:USB Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:USB:BASE <base> (see page 1139)	:SBUS<n>:USB:BASE? (see page 1139)	<base> ::= {ASCIi BINary DECimal HEX}
:SBUS<n>:USB:SOURce:D MINus <source> (see page 1140)	:SBUS<n>:USB:SOURce:D MINus? (see page 1140)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:USB:SOURce:D PLus <source> (see page 1141)	:SBUS<n>:USB:SOURce:D PLus? (see page 1141)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:USB:SOURce:D IFFerential <source> (see page 1142)	:SBUS<n>:USB:SOURce:D IFFerential? (see page 1142)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:USB:SPEEd <speed> (see page 1143)	:SBUS<n>:USB:SPEEd? (see page 1143)	<speed> ::= {LOW FULL HIGH}
:SBUS<n>:USB:TRIGger <condition> (see page 1144)	:SBUS<n>:USB:TRIGger? (see page 1144)	<condition> ::= {SOP EOP ENTersuspend EXITSuspend RESet TOKEen DATA HANDshake SPECial ALLerrors PIDerror CRC5error CRC16error GLITCherror STUFFerror SEIerror}

Table 40 :SBUS<n>:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:USB:TRIGger:ADDRESS <string> (see page 1145)	:SBUS<n>:USB:TRIGger:ADDRESS? (see page 1145)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:CRC <string> (see page 1146)	:SBUS<n>:USB:TRIGger:CRC? (see page 1146)	<string> ::= "nnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:DATA <string> (see page 1147)	:SBUS<n>:USB:TRIGger:DATA? (see page 1147)	<string> ::= "nnnn..." where n ::= {0 1 X} <string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:DATA:LENGth <value> (see page 1148)	:SBUS<n>:USB:TRIGger:DATA:LENGth? (see page 1148)	<length> ::= data length between 1-20
:SBUS<n>:USB:TRIGger:ENDPoint <string> (see page 1149)	:SBUS<n>:USB:TRIGger:ENDPoint? (see page 1149)	<string> ::= "nnnn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:ET <string> (see page 1150)	:SBUS<n>:USB:TRIGger:ET? (see page 1150)	<string> ::= "nn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0 1 2 3 X \$}
:SBUS<n>:USB:TRIGger:FRaME <string> (see page 1151)	:SBUS<n>:USB:TRIGger:FRaME? (see page 1151)	<string> ::= "nnnnnnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnnnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:HADDRESS <string> (see page 1152)	:SBUS<n>:USB:TRIGger:HADDRESS? (see page 1152)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:PCHeck <string> (see page 1153)	:SBUS<n>:USB:TRIGger:PCHeck? (see page 1153)	<string> ::= "nnnn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:PID:DATA <pid> (see page 1154)	:SBUS<n>:USB:TRIGger:PID:DATA? (see page 1154)	<pid> ::= {DATA0 DATA1 DATA2 MDATa}

Table 40 :SBUS<n>:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:USB:TRIGger:PID:HANDshake <pid> (see page 1155)	:SBUS<n>:USB:TRIGger:PID:HANDshake? (see page 1155)	<pid> ::= {ACK NAK STALL NYET}
:SBUS<n>:USB:TRIGger:PID:SPECIAL <pid> (see page 1156)	:SBUS<n>:USB:TRIGger:PID:SPECIAL? (see page 1156)	<pid> ::= {PING PRE ERR SPLit}
:SBUS<n>:USB:TRIGger:PID:TOKEN <pid> (see page 1157)	:SBUS<n>:USB:TRIGger:PID:TOKEN? (see page 1157)	<pid> ::= {OUT IN SETUP SOF}
:SBUS<n>:USB:TRIGger:PORT <string> (see page 1158)	:SBUS<n>:USB:TRIGger:PORT? (see page 1158)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:SC <string> (see page 1159)	:SBUS<n>:USB:TRIGger:SC? (see page 1159)	<string> ::= "n" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0 1 X \$}
:SBUS<n>:USB:TRIGger:SEU <string> (see page 1160)	:SBUS<n>:USB:TRIGger:SEU? (see page 1160)	<string> ::= "nn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0 1 2 3 X \$}

Table 41 :SBUS<n>:USBPD Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:USBPD:SOURce <source> (see page 1162)	:SBUS<n>:USBPD:SOURce ? (see page 1162)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:USBPD:TRIGger <mode> (see page 1163)	:SBUS<n>:USBPD:TRIGger? (see page 1163)	<mode> ::= {PSTart EOP SOP SPRime SDPRime SPDebug SDPDebug HRST CRST CRCerror PERRor HEADER}
:SBUS<n>:USBPD:TRIGger:HEADER <type> (see page 1164)	:SBUS<n>:USBPD:TRIGger:HEADER? (see page 1164)	<type> ::= {CMESsage DMESsage EMESsage VALue}

Table 41 :SBUS<n>:USBPd Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage <type> (see page 1166)	:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage? (see page 1166)	<type> ::= {GOODCrc GOTOMin ACCEpt REJect PING PSRDy GSRCap GSNCap DRSSwap PRSSwap VCSwap WAIT SRST GSCX GSTatus FRSSwap GPSTatus GCCodes}
:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage <type> (see page 1168)	:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage? (see page 1168)	<type> ::= {SRCap REQuest BIST SNCap BSTatus ALERt GCINfo VDEfined}
:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage <type> (see page 1169)	:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage? (see page 1169)	<type> ::= {SCX STATus GBCap GBSTatus BCAP GMINfo MINfo SREQuest SRESponse FREQuest FRESponse PSTatus CINfo CCODEs}
:SBUS<n>:USBPd:TRIGger:HEADer:VALue <string> (see page 1171)	:SBUS<n>:USBPd:TRIGger:HEADer:VALue? (see page 1171)	<string> ::= "nn...n" where n ::= {0 1 X} <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X}
:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier <type> (see page 1172)	:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier? (see page 1172)	<type> ::= {NONE SOP SPRime SDPRime}

Table 42 General :SEARCh Commands Summary

Command	Query	Options and Query Returns
n/a	:SEARCh:COUNT? (see page 1175)	<count> ::= an integer count value
:SEARCh:EVENT <event_number> (see page 1176)	:SEARCh:EVENT? (see page 1176)	<event_number> ::= the integer number of a found search event
:SEARCh:MODE <value> (see page 1177)	:SEARCh:MODE? (see page 1177)	<value> ::= {EDGE GLITCh RUNT TRANSition SERIAL{1 2} PEAK}
:SEARCh:STATe <value> (see page 1178)	:SEARCh:STATe? (see page 1178)	<value> ::= {{0 OFF} {1 ON}}

Table 43 :SEARCH:EDGE Commands Summary

Command	Query	Options and Query Returns
:SEARCH:EDGE:SLOPe <slope> (see page 1180)	:SEARCH:EDGE:SLOPe? (see page 1180)	<slope> ::= {POSitive NEGative EITHER}
:SEARCH:EDGE:SOURce <source> (see page 1181)	:SEARCH:EDGE:SOURce? (see page 1181)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

Table 44 :SEARCH:GLITCh Commands Summary

Command	Query	Options and Query Returns
:SEARCH:GLITCh:GREate rthan <greater_than_time>[s uffix] (see page 1183)	:SEARCH:GLITCh:GREate rthan? (see page 1183)	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:SEARCH:GLITCh:LESSth an <less_than_time>[suff ix] (see page 1184)	:SEARCH:GLITCh:LESSth an? (see page 1184)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:SEARCH:GLITCh:POLari ty <polarity> (see page 1185)	:SEARCH:GLITCh:POLari ty? (see page 1185)	<polarity> ::= {POSitive NEGative}
:SEARCH:GLITCh:QUALif ier <qualifier> (see page 1186)	:SEARCH:GLITCh:QUALif ier? (see page 1186)	<qualifier> ::= {GREaterthan LESSthan RANGE}
:SEARCH:GLITCh:RANGe <less_than_time>[suff ix], <greater_than_time>[s uffix] (see page 1187)	:SEARCH:GLITCh:RANGe? (see page 1187)	<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 page 1188)	:SEARCH:GLITCh:SOURce ? (see page 1188)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

Table 45 :SEARCH:PEAK Commands Summary

Command	Query	Options and Query Returns
:SEARCH:PEAK:EXCURSION <delta_level> (see page 1190)	:SEARCH:PEAK:EXCURSION? (see page 1190)	<delta_level> ::= required change in level to be recognized as a peak, in NR3 format.
:SEARCH:PEAK:NPEAKS <number> (see page 1191)	:SEARCH:PEAK:NPEAKS? (see page 1191)	<number> ::= max number of peaks to find, 1-11 in NR1 format.
:SEARCH:PEAK:SOURCE <source> (see page 1192)	:SEARCH:PEAK:SOURCE? (see page 1192)	<source> ::= {FUNCTION<m> MATH<m>} (must be an FFT waveform) <m> ::= 1 to 4 in NR1 format
:SEARCH:PEAK:THRESHOLD <level> (see page 1193)	:SEARCH:PEAK:THRESHOLD? (see page 1193)	<level> ::= necessary level to be considered a peak, in NR3 format.

Table 46 :SEARCH:RUNT Commands Summary

Command	Query	Options and Query Returns
:SEARCH:RUNT:POLARITY <polarity> (see page 1195)	:SEARCH:RUNT:POLARITY? (see page 1195)	<polarity> ::= {POSITIVE NEGATIVE EITHER}
:SEARCH:RUNT:QUALIFIER <qualifier> (see page 1196)	:SEARCH:RUNT:QUALIFIER? (see page 1196)	<qualifier> ::= {GREATERthan LESSthan NONE}
:SEARCH:RUNT:SOURCE <source> (see page 1197)	:SEARCH:RUNT:SOURCE? (see page 1197)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARCH:RUNT:TIME <time>[suffix] (see page 1198)	:SEARCH:RUNT:TIME? (see page 1198)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 47 :SEARCh:TRANSition Commands Summary

Command	Query	Options and Query Returns
:SEARCh:TRANSition:QU ALifier <qualifier> (see page 1200)	:SEARCh:TRANSition:QU ALifier? (see page 1200)	<qualifier> ::= {GREaterthan LESSthan}
:SEARCh:TRANSition:SL OPe <slope> (see page 1201)	:SEARCh:TRANSition:SL OPe? (see page 1201)	<slope> ::= {NEGative POSitive}
:SEARCh:TRANSition:SO URce <source> (see page 1202)	:SEARCh:TRANSition:SO URce? (see page 1202)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARCh:TRANSition:TI ME <time>[suffix] (see page 1203)	:SEARCh:TRANSition:TI ME? (see page 1203)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 48 :SEARCh:SERial:A429 Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:A429:L ABel <value> (see page 1205)	:SEARCh:SERial:A429:L ABel? (see page 1205)	<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 page 1206)	:SEARCh:SERial:A429:M ODE? (see page 1206)	<condition> ::= {LABel LBITs PERRor WERRor GERRor WGERrors ALLerrors}
:SEARCh:SERial:A429:P ATtern:DATA <string> (see page 1207)	:SEARCh:SERial:A429:P ATtern:DATA? (see page 1207)	<string> ::= "nn...n" where n ::= {0 1}, length depends on FORMat
:SEARCh:SERial:A429:P ATtern:SDI <string> (see page 1208)	:SEARCh:SERial:A429:P ATtern:SDI? (see page 1208)	<string> ::= "nn" where n ::= {0 1}, length always 2 bits
:SEARCh:SERial:A429:P ATtern:SSM <string> (see page 1209)	:SEARCh:SERial:A429:P ATtern:SSM? (see page 1209)	<string> ::= "nn" where n ::= {0 1}, length always 2 bits

Table 49 :SEARCH:SERIAL:CAN Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:CAN:MODE <value> (see page 1211)	:SEARCH:SERIAL:CAN:MODE? (see page 1211)	<value> ::= {IDEither IDData DATA IDRemote ERROR ACKError FORMError STUFFerror CRCError ALLerrors OVERload MESSAGE MSIGNAL}
:SEARCH:SERIAL:CAN:PA TTern:DATA <string> (see page 1213)	:SEARCH:SERIAL:CAN:PA TTern:DATA? (see page 1213)	<string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X} for hexadecimal
:SEARCH:SERIAL:CAN:PA TTern:DATA:LENGTH <length> (see page 1214)	:SEARCH:SERIAL:CAN:PA TTern:DATA:LENGTH? (see page 1214)	<length> ::= integer from 1 to 8 in NR1 format
:SEARCH:SERIAL:CAN:PA TTern:ID <string> (see page 1215)	:SEARCH:SERIAL:CAN:PA TTern:ID? (see page 1215)	<string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X} for hexadecimal
:SEARCH:SERIAL:CAN:PA TTern:ID:MODE <value> (see page 1216)	:SEARCH:SERIAL:CAN:PA TTern:ID:MODE? (see page 1216)	<value> ::= {STANDARD EXTENDED}
:SEARCH:SERIAL:CAN:SY MBolic:MESSAGE <name> (see page 1217)	:SEARCH:SERIAL:CAN:SY MBolic:MESSAGE? (see page 1217)	<name> ::= quoted ASCII string
:SEARCH:SERIAL:CAN:SY MBolic:SIGNAL <name> (see page 1218)	:SEARCH:SERIAL:CAN:SY MBolic:SIGNAL? (see page 1218)	<name> ::= quoted ASCII string
:SEARCH:SERIAL:CAN:SY MBolic:VALUE <data> (see page 1219)	:SEARCH:SERIAL:CAN:SY MBolic:VALUE? (see page 1219)	<data> ::= value in NR3 format

Table 50 :SEARCH:SERIAL:IIC Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:IIC:MODE <value> (see page 1221)	:SEARCH:SERIAL:IIC:MODE? (see page 1221)	<value> ::= {REStart ADDRESS ANACK NACKnowledge REPRom READ7 WRITE7 R7Data2 W7Data2}
:SEARCH:SERIAL:IIC:PA TTern:ADDRESS <value> (see page 1223)	:SEARCH:SERIAL:IIC:PA TTern:ADDRESS? (see page 1223)	<value> ::= integer
:SEARCH:SERIAL:IIC:PA TTern:DATA <value> (see page 1224)	:SEARCH:SERIAL:IIC:PA TTern:DATA? (see page 1224)	<value> ::= integer

Table 50 :SEARCH:SERIAL:IIC Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARCH:SERIAL:IIC:PA TTern:DATA2 <value> (see page 1225)	:SEARCH:SERIAL:IIC:PA TTern:DATA2? (see page 1225)	<value> ::= integer
:SEARCH:SERIAL:IIC:QU ALifier <value> (see page 1226)	:SEARCH:SERIAL:IIC:QU ALifier? (see page 1226)	<value> ::= {EQUAL NOTequal LESSthan GREATERthan}

Table 51 :SEARCH:SERIAL:LIN Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:LIN:ID <value> (see page 1228)	:SEARCH:SERIAL:LIN:ID ? (see page 1228)	<value> ::= 7-bit integer in decimal or <nondecimal> from 0-63 <nondecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 1} for binary
:SEARCH:SERIAL:LIN:MO DE <value> (see page 1229)	:SEARCH:SERIAL:LIN:MO DE? (see page 1229)	<value> ::= {ID DATA ERRor}
:SEARCH:SERIAL:LIN:PA TTern:DATA <string> (see page 1230)	:SEARCH:SERIAL:LIN:PA TTern:DATA? (see page 1230)	When :SEARCH:SERIAL:LIN:PATtern:FORMa t DECimal, <string> ::= "n" where n ::= 32-bit integer in unsigned decimal, returns "\$" if data has any don't cares When :SEARCH:SERIAL:LIN:PATtern:FORMa t HEX, <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X }
:SEARCH:SERIAL:LIN:PA TTern:DATA:LENGth <length> (see page 1231)	:SEARCH:SERIAL:LIN:PA TTern:DATA:LENGth? (see page 1231)	<length> ::= integer from 1 to 8 in NR1 format
:SEARCH:SERIAL:LIN:PA TTern:FORMat <base> (see page 1232)	:SEARCH:SERIAL:LIN:PA TTern:FORMat? (see page 1232)	<base> ::= {HEX DECimal}
:SEARCH:SERIAL:LIN:SY MBolic:FRAME <name> (see page 1233)	:SEARCH:SERIAL:LIN:SY MBolic:FRAME? (see page 1233)	<name> ::= quoted ASCII string

Table 51 :SEARCh:SErIAL:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARCh:SErIAL:LIN:SYMBolic:SIGNAL <name> (see page 1234)	:SEARCh:SErIAL:LIN:SYMBolic:SIGNAL? (see page 1234)	<name> ::= quoted ASCII string
:SEARCh:SErIAL:LIN:SYMBolic:VALue <data> (see page 1235)	:SEARCh:SErIAL:LIN:SYMBolic:VALue? (see page 1235)	<data> ::= value in NR3 format

Table 52 :SEARCh:SErIAL:M1553 Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SErIAL:M1553:MODE <value> (see page 1237)	:SEARCh:SErIAL:M1553:MODE? (see page 1237)	<value> ::= {DStart CStart RTA RTA11 PERRor SERRor MERRor}
:SEARCh:SErIAL:M1553:PATtern:DATA <string> (see page 1238)	:SEARCh:SErIAL:M1553:PATtern:DATA? (see page 1238)	<string> ::= "nn...n" where n ::= {0 1}
:SEARCh:SErIAL:M1553:RTA <value> (see page 1239)	:SEARCh:SErIAL:M1553:RTA? (see page 1239)	<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 53 :SEARCh:SErIAL:SENT Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SErIAL:SENT:FAST:DATA <string> (see page 1241)	:SEARCh:SErIAL:SENT:FAST:DATA? (see page 1241)	<string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SEARCh:SErIAL:SENT:MODE <mode> (see page 1242)	:SEARCh:SErIAL:SENT:MODE? (see page 1242)	<mode> ::= {FCData SCMid SCData CRCerror PPERror}

Table 53 :SEARCH:SERIAL:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARCH:SERIAL:SENT:SLow:DATA <data> (see page 1243)	:SEARCH:SERIAL:SENT:SLow:DATA? (see page 1243)	<data> ::= from -1 (don't care) to 65535, in NR1 format.
:SEARCH:SERIAL:SENT:SLow:ID <id> (see page 1244)	:SEARCH:SERIAL:SENT:SLow:ID? (see page 1244)	<id> ::= from -1 (don't care) to 255, in NR1 format.

Table 54 :SEARCH:SERIAL:SPI Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:SPI:MODE <value> (see page 1246)	:SEARCH:SERIAL:SPI:MODE? (see page 1246)	<value> ::= {MOSI MISO}
:SEARCH:SERIAL:SPI:PARTern:DATA <string> (see page 1247)	:SEARCH:SERIAL:SPI:PARTern:DATA? (see page 1247)	<string> ::= "0xn...n" where n ::= {0,...,9 A,...,F X}
:SEARCH:SERIAL:SPI:PARTern:WIDTH <width> (see page 1248)	:SEARCH:SERIAL:SPI:PARTern:WIDTH? (see page 1248)	<width> ::= integer from 1 to 10

Table 55 :SEARCH:SERIAL:UART Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:UART:DATA <value> (see page 1250)	:SEARCH:SERIAL:UART:DATA? (see page 1250)	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, or <binary> format <hexadecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0 1} for binary
:SEARCH:SERIAL:UART:MODE <value> (see page 1251)	:SEARCH:SERIAL:UART:MODE? (see page 1251)	<value> ::= {RDATa RD1 RD0 RDX TDATa TD1 TD0 TDX RXParity TXParity AERRor}
:SEARCH:SERIAL:UART:QUALifier <value> (see page 1253)	:SEARCH:SERIAL:UART:QUALifier? (see page 1253)	<value> ::= {EQUal NOTequal GREaterthan LESSthan}

Table 56 :SEARCH:SERIAL:USB Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:USB:MODE <condition> (see page 1256)	:SEARCH:SERIAL:USB:MODE? (see page 1256)	<event> ::= {Token DATA HANDshake SPECIAL ALLerrors PIDerror CRC5error CRC16error GLITCherror STUFFerror SEIerror}
:SEARCH:SERIAL:USB:ADDRESS <string> (see page 1257)	:SEARCH:SERIAL:USB:ADDRESS? (see page 1257)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:CRCC <string> (see page 1258)	:SEARCH:SERIAL:USB:CRCC? (see page 1258)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:DATA <string> (see page 1259)	:SEARCH:SERIAL:USB:DATA? (see page 1259)	<string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:DATA:LENGTH <value> (see page 1260)	:SEARCH:SERIAL:USB:DATA:LENGTH? (see page 1260)	<length> ::= data length between 1-20
:SEARCH:SERIAL:USB:ENDPOINT <string> (see page 1261)	:SEARCH:SERIAL:USB:ENDPOINT? (see page 1261)	<string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:ET <string> (see page 1262)	:SEARCH:SERIAL:USB:ET? (see page 1262)	<string> ::= "0xn" where n ::= {0 1 2 3 X \$}
:SEARCH:SERIAL:USB:FRAME <string> (see page 1263)	:SEARCH:SERIAL:USB:FRAME? (see page 1263)	<string> ::= "0xnnnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:HANDADDRESS <string> (see page 1264)	:SEARCH:SERIAL:USB:HANDADDRESS? (see page 1264)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:PID:DATA <pid> (see page 1265)	:SEARCH:SERIAL:USB:PID:DATA? (see page 1265)	<pid> ::= {DATA0 DATA1 DATA2 MDATA}
:SEARCH:SERIAL:USB:PID:HANDshake <pid> (see page 1266)	:SEARCH:SERIAL:USB:PID:HANDshake? (see page 1266)	<pid> ::= {ACK NAK STALL NYET}
:SEARCH:SERIAL:USB:PID:SPECIAL <pid> (see page 1267)	:SEARCH:SERIAL:USB:PID:SPECIAL? (see page 1267)	<pid> ::= {PING PRE ERR SPLIT}
:SEARCH:SERIAL:USB:PID:TOKEN <pid> (see page 1268)	:SEARCH:SERIAL:USB:PID:TOKEN? (see page 1268)	<pid> ::= {OUT IN SETUP SOF}

Table 56 :SEARCH:SERIAL:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARCH:SERIAL:USB:PORT <string> (see page 1269)	:SEARCH:SERIAL:USB:PORT? (see page 1269)	<string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:SC <string> (see page 1270)	:SEARCH:SERIAL:USB:SC? (see page 1270)	<string> ::= "0xn" where n ::= {0 1 X \$}
:SEARCH:SERIAL:USB:SEU <string> (see page 1271)	:SEARCH:SERIAL:USB:SEU? (see page 1271)	<string> ::= "0xn" where n ::= {0 1 2 3 X \$}

Table 57 :SEARCH:SERIAL:USBPD Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:USBPD:MODE <mode> (see page 1273)	:SEARCH:SERIAL:USBPD:MODE? (see page 1273)	<mode> ::= {SOP SPrime SDPrime SPDebug SDPDebug HRST CRST CRCError PERRor HEADer}
:SEARCH:SERIAL:USBPD:HEADer <type> (see page 1274)	:SEARCH:SERIAL:USBPD:HEADer? (see page 1274)	<type> ::= {CMESsage DMESsage EMESsage VALue}
:SEARCH:SERIAL:USBPD:HEADer:CMESsage <type> (see page 1276)	:SEARCH:SERIAL:USBPD:HEADer:CMESsage? (see page 1276)	<type> ::= {GOODcrc GOTomin ACCEpt REJect PING PSRDy GSRCap GSNCap DRSWap PRSWap VCSWap WAIT SRST GSCX GSTatus FRSWap GPSTatus GCCodes}
:SEARCH:SERIAL:USBPD:HEADer:DMESsage <type> (see page 1278)	:SEARCH:SERIAL:USBPD:HEADer:DMESsage? (see page 1278)	<type> ::= {SRCap REQuest BIST SNCap BSTatus ALERt GCInfo VDEfined}
:SEARCH:SERIAL:USBPD:HEADer:EMESsage <type> (see page 1279)	:SEARCH:SERIAL:USBPD:HEADer:EMESsage? (see page 1279)	<type> ::= {SCX STATus GBCap GBSTatus BCAP GMINfo MINfo SREQuest SRESponse FREQuest FRESponse PSTatus CINfo CCODEs}
:SEARCH:SERIAL:USBPD:HEADer:QUALifier <type> (see page 1281)	:SEARCH:SERIAL:USBPD:HEADer:QUALifier? (see page 1281)	<type> ::= {NONE SOP SPrime SDPrime}
:SEARCH:SERIAL:USBPD:HEADer:VALue <string> (see page 1282)	:SEARCH:SERIAL:USBPD:HEADer:VALue? (see page 1282)	<string> ::= "nn...n" where n ::= {0 1 X} <string> ::= "0xn...n" where n ::= {0,...,9 A,...,F X \$}

Table 58 General :STATus Commands Summary

Command	Query	Options and Query Returns
:STATus:PRESet (see page 1287)	n/a	n/a

Table 59 :STATus:OPERation Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:BIT :CONDition? (see page 1293)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :ENABle <bit_value> (see page 1294)	:STATus:OPERation:BIT :ENABle? (see page 1294)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :NTRansition <bit_value> (see page 1295)	:STATus:OPERation:BIT :NTRansition? (see page 1295)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :PTRansition <bit_value> (see page 1296)	:STATus:OPERation:BIT :PTRansition? (see page 1296)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:BIT [:EVENT]? (see page 1297)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:CON Dition? (see page 1298)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ENA Ble <sum_of_bits> (see page 1299)	:STATus:OPERation:ENA Ble? (see page 1299)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:NTR ansition <sum_of_bits> (see page 1300)	:STATus:OPERation:NTR ansition? (see page 1300)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 59 :STATus:OPERation Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:PTR ansition <sum_of_bits> (see page 1301)	:STATus:OPERation:PTR ansition? (see page 1301)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation[:EV ENT]? (see page 1302)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 60 :STATus:OPERation:ARM Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:ARM :BIT:CONDition? (see page 1306)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:ENABle <bit_value> (see page 1307)	:STATus:OPERation:ARM :BIT:ENABle? (see page 1307)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:NTRansition <bit_value> (see page 1308)	:STATus:OPERation:ARM :BIT:NTRansition? (see page 1308)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:PTRansition <bit_value> (see page 1309)	:STATus:OPERation:ARM :BIT:PTRansition? (see page 1309)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM :BIT[:EVENT]? (see page 1310)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM :CONDition? (see page 1311)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :ENABle <sum_of_bits> (see page 1312)	:STATus:OPERation:ARM :ENABle? (see page 1312)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 60 :STATus:OPERation:ARM Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:ARM :NTRansition <sum_of_bits> (see page 1313)	:STATus:OPERation:ARM :NTRansition? (see page 1313)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :PTRansition <sum_of_bits> (see page 1314)	:STATus:OPERation:ARM :PTRansition? (see page 1314)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM [:EVENT]? (see page 1315)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 61 :STATus:OPERation:HARDware Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:HAR Dware:BIT:CONDitio n? (see page 1319)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HAR Dware:BIT:ENABLE <bit_value> (see page 1320)	:STATus:OPERation:HAR Dware:BIT:ENABLE? (see page 1320)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HAR Dware:BIT:NTRansit ion <bit_value> (see page 1321)	:STATus:OPERation:HAR Dware:BIT:NTRansit ion? (see page 1321)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HAR Dware:BIT:PTRansit ion <bit_value> (see page 1322)	:STATus:OPERation:HAR Dware:BIT:PTRansit ion? (see page 1322)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HAR Dware:BIT[:EVENT]? (see page 1323)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HAR Dware:CONDition? (see page 1324)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 61 :STATus:OPERation:HARDware Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:HARDware:ENABle <sum_of_bits> (see page 1325)	:STATus:OPERation:HARDware:ENABle? (see page 1325)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:NTRansition <sum_of_bits> (see page 1326)	:STATus:OPERation:HARDware:NTRansition? (see page 1326)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:PTRansition <sum_of_bits> (see page 1327)	:STATus:OPERation:HARDware:PTRansition? (see page 1327)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HARDware[:EVENT]? (see page 1328)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 62 :STATus:OPERation:LOCal Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:LOCAl:BIT:CONDition? (see page 1332)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:ENABle <bit_value> (see page 1333)	:STATus:OPERation:LOCAl:BIT:ENABle? (see page 1333)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:NTRansition <bit_value> (see page 1334)	:STATus:OPERation:LOCAl:BIT:NTRansition? (see page 1334)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:PTRansition <bit_value> (see page 1335)	:STATus:OPERation:LOCAl:BIT:PTRansition? (see page 1335)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:LOCAl:BIT[:EVENT]? (see page 1336)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format

Table 62 :STATus:OPERation:LOCal Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:LOC al:CONDition? (see page 1337)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOC al:ENABle <sum_of_bits> (see page 1338)	:STATus:OPERation:LOC al:ENABle? (see page 1338)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOC al:NTRansition <sum_of_bits> (see page 1339)	:STATus:OPERation:LOC al:NTRansition? (see page 1339)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOC al:PTRansition <sum_of_bits> (see page 1340)	:STATus:OPERation:LOC al:PTRansition? (see page 1340)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:LOC al[:EVENT]? (see page 1341)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 63 :STATus:OPERation:MTESt Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:MTE St:BIT:CONDition? (see page 1345)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTE St:BIT:ENABle <bit_value> (see page 1346)	:STATus:OPERation:MTE St:BIT:ENABle? (see page 1346)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTE St:BIT:NTRansition <bit_value> (see page 1347)	:STATus:OPERation:MTE St:BIT:NTRansition ? (see page 1347)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTE St:BIT:PTRansition <bit_value> (see page 1348)	:STATus:OPERation:MTE St:BIT:PTRansition ? (see page 1348)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format

Table 63 :STATus:OPERation:MTESt Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:MTESt:BIT[:EVENT]? (see page 1349)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:MTESt:CONDition? (see page 1350)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:ENABle <sum_of_bits> (see page 1351)	:STATus:OPERation:MTESt:ENABle? (see page 1351)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:NTRansition <sum_of_bits> (see page 1352)	:STATus:OPERation:MTESt:NTRansition? (see page 1352)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:PTRansition <sum_of_bits> (see page 1353)	:STATus:OPERation:MTESt:PTRansition? (see page 1353)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:MTESt[:EVENT]? (see page 1354)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 64 :STATus:OPERation:OVERload Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:OVERload:BIT:CONDition? (see page 1358)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:BIT:ENABle <bit_value> (see page 1359)	:STATus:OPERation:OVERload:BIT:ENABle? (see page 1359)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:BIT:NTRansition <bit_value> (see page 1360)	:STATus:OPERation:OVERload:BIT:NTRansition? (see page 1360)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format

Table 64 :STATus:OPERation:OVERload Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:OVERload:BIT:PTRansition <bit_value> (see page 1361)	:STATus:OPERation:OVERload:BIT:PTRansition? (see page 1361)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:BIT[:EVENT]? (see page 1362)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:CONDition? (see page 1363)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:ENABle <sum_of_bits> (see page 1364)	:STATus:OPERation:OVERload:ENABle? (see page 1364)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:NTRansition <sum_of_bits> (see page 1365)	:STATus:OPERation:OVERload:NTRansition? (see page 1365)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PTRansition <sum_of_bits> (see page 1366)	:STATus:OPERation:OVERload:PTRansition? (see page 1366)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload[:EVENT]? (see page 1367)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 65 :STATus:OPERation:OVERload:PFAult Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:OVERload:PFAult:BIT:CONDition? (see page 1371)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:BIT:ENABle <bit_value> (see page 1372)	:STATus:OPERation:OVERload:PFAult:BIT:ENABle? (see page 1372)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format

Table 65 :STATus:OPERation:OVERload:PFAult Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:OVERload:PFAult:BIT:NTRansition <bit_value> (see page 1373)	:STATus:OPERation:OVERload:PFAult:BIT:NTRansition? (see page 1373)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:BIT:PTRansition <bit_value> (see page 1374)	:STATus:OPERation:OVERload:PFAult:BIT:PTRansition? (see page 1374)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFAult:BIT[:EVENT]? (see page 1375)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFAult:CONDition? (see page 1376)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:ENABLE <sum_of_bits> (see page 1377)	:STATus:OPERation:OVERload:PFAult:ENABLE? (see page 1377)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:NTRansition <sum_of_bits> (see page 1378)	:STATus:OPERation:OVERload:PFAult:NTRansition? (see page 1378)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:PTRansition <sum_of_bits> (see page 1379)	:STATus:OPERation:OVERload:PFAult:PTRansition? (see page 1379)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFAult[:EVENT]? (see page 1380)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 66 :STATus:OPERation:POWer Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:POW er:BIT:CONDition? (see page 1384)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:BIT:ENABle <bit_value> (see page 1385)	:STATus:OPERation:POW er:BIT:ENABle? (see page 1385)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:BIT:NTRansition <bit_value> (see page 1386)	:STATus:OPERation:POW er:BIT:NTRansition ? (see page 1386)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:BIT:PTRansition <bit_value> (see page 1387)	:STATus:OPERation:POW er:BIT:PTRansition ? (see page 1387)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POW er:BIT[:EVENT]? (see page 1388)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POW er:CONDition? (see page 1389)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:ENABle <sum_of_bits> (see page 1390)	:STATus:OPERation:POW er:ENABle? (see page 1390)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:NTRansition <sum_of_bits> (see page 1391)	:STATus:OPERation:POW er:NTRansition? (see page 1391)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:POW er:PTRansition <sum_of_bits> (see page 1392)	:STATus:OPERation:POW er:PTRansition? (see page 1392)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POW er[:EVENT]? (see page 1393)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 67 :STATus:TRIGger Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:TRIGger:BIT:CONDition? (see page 1397)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:ENABle <bit_value> (see page 1398)	:STATus:TRIGger:BIT:ENABle? (see page 1398)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:NTRansition <bit_value> (see page 1399)	:STATus:TRIGger:BIT:NTRansition? (see page 1399)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:PTRansition <bit_value> (see page 1400)	:STATus:TRIGger:BIT:PTRansition? (see page 1400)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger:BIT[:EVENT]? (see page 1401)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger:CONDition? (see page 1402)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:TRIGger:ENABle <sum_of_bits> (see page 1403)	:STATus:TRIGger:ENABle? (see page 1403)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:TRIGger:NTRansition <sum_of_bits> (see page 1404)	:STATus:TRIGger:NTRansition? (see page 1404)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:TRIGger:PTRansition <sum_of_bits> (see page 1405)	:STATus:TRIGger:PTRansition? (see page 1405)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger[:EVENT]? (see page 1406)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 68 :STATus:USER Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:USER:BIT:CONDition? (see page 1410)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:ENABLE <bit_value> (see page 1411)	:STATus:USER:BIT:ENABLE? (see page 1411)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:NTransition <bit_value> (see page 1412)	:STATus:USER:BIT:NTransition? (see page 1412)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:PTransition <bit_value> (see page 1413)	:STATus:USER:BIT:PTransition? (see page 1413)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:USER:BIT[:EVENT]? (see page 1414)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:USER:CONDition? (see page 1415)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:USER:ENABLE <sum_of_bits> (see page 1416)	:STATus:USER:ENABLE? (see page 1416)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:USER:NTransit ion <sum_of_bits> (see page 1417)	:STATus:USER:NTransit ion? (see page 1417)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:USER:PTransit ion <sum_of_bits> (see page 1418)	:STATus:USER:PTransit ion? (see page 1418)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:USER[:EVENT]? (see page 1419)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 69 :SYSTem Commands Summary

Command	Query	Options and Query Returns
:SYSTem:DATE <date> (see page 1425)	:SYSTem:DATE? (see page 1425)	<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}
n/a	:SYSTem:DIIdentifier? (see page 1426)	n/a
:SYSTem:DSP <string> (see page 1427)	n/a	<string> ::= up to 75 characters as a quoted ASCII string
n/a	:SYSTem:ERROR:ALL? (see page 1428)	<errors_list> ::= comma-separated list of all errors from the queue (codes and strings)
n/a	:SYSTem:ERROR:CODE[:NEXT]? (see page 1429)	<error_code> ::= the next error (code only) from the queue
n/a	:SYSTem:ERROR:COUNT? (see page 1430)	<count> ::= an integer number of errors in the queue
n/a	:SYSTem:ERROR:EXTended? <type> (see page 1431)	<type> ::= {INFO WARNING ERROR FATAL MESSAGE ALL} <errors_list> ::= returns a comma-separated list of all errors (strings, no codes) of the specified type from the queue
n/a	:SYSTem:ERROR[:NEXT]? (see page 1432)	<error_number>,<error_string> ::= the next error (code and string) from the queue <error_number> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see page 1661).
n/a	:SYSTem:HELP:HEADers? (see page 1434)	<binary_block> ::= newline-separated list of supported SCPI commands/queries
n/a	:SYSTem:HID? (see page 1435)	n/a

Table 69 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:LOCK <value> (see page 1436)	:SYSTem:LOCK? (see page 1436)	<value> ::= {{1 ON} {0 OFF}}
n/a	:SYSTem:LOCK:OWNer? (see page 1437)	<session_string> ::= quoted session string or "NONE"
:SYSTem:LOCK:RELease (see page 1438)	n/a	n/a
n/a	:SYSTem:LOCK:REQuest? (see page 1439)	{1 0} (for granted or not)
:SYSTem:PERSon[:MANu facturer] <manufacturer_string> (see page 1440)	:SYSTem:PERSon[:MANu facturer]? (see page 1440)	<manufacturer_string> ::= quoted ASCII string, up to 63 characters
:SYSTem:PERSon[:MANu facturer]:DEfault (see page 1441)	n/a	Sets manufacturer string to "KEYSIGHT TECHNOLOGIES"
:SYSTem:POWercycle (see page 1442)	n/a	n/a
:SYSTem:PRECision:LEN Gth <length> (see page 1443)	:SYSTem:PRECision:LEN Gth? (see page 1443)	<length> ::= {64K 128K 256K 512K 1M 1.5M 2M 4M 8M 12M 16M 24M 32M}
:SYSTem:PRECision:PEA K {{0 OFF} {1 ON}} (see page 1444)	:SYSTem:PRECision:PEA K? (see page 1444)	<setting> ::= {0 1}
:SYSTem:PRESet (see page 1445)	n/a	See :SYSTem:PRESet (see page 1445)
:SYSTem:PRODuction:FR ESet (see page 1445)	n/a	See :SYSTem:PRODuction:FRESet (see page 1448)
:SYSTem:PROTection:LO CK <value> (see page 1449)	:SYSTem:PROTection:LO CK? (see page 1449)	<value> ::= {{1 ON} {0 OFF}}
:SYSTem:RLOGger <setting>[,<file_name>[,<write_mode>]] (see page 1450)	n/a	<setting> ::= {{0 OFF} {1 ON}} <file_name> ::= quoted ASCII string <write_mode> ::= {CREate APPend}
:SYSTem:RLOGger:DESTi nation <dest> (see page 1451)	:SYSTem:RLOGger:DESTi nation? (see page 1451)	<dest> ::= {FILE SCReen BOTH}

Table 69 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:RLOGger:DISPlay {{0 OFF} {1 ON}} (see page 1452)	:SYSTem:RLOGger:DISPlay? (see page 1452)	<setting> ::= {0 1}
:SYSTem:RLOGger:FNAME <file_name> (see page 1453)	:SYSTem:RLOGger:FNAME? (see page 1453)	<file_name> ::= quoted ASCII string
:SYSTem:RLOGger:STATe {{0 OFF} {1 ON}} (see page 1454)	:SYSTem:RLOGger:STATe? (see page 1454)	<setting> ::= {0 1}
:SYSTem:RLOGger:TRANs parent {{0 OFF} {1 ON}} (see page 1455)	:SYSTem:RLOGger:TRANs parent? (see page 1455)	<setting> ::= {0 1}
:SYSTem:RLOGger:WMODE <write_mode> (see page 1456)	:SYSTem:RLOGger:WMODE? (see page 1456)	<write_mode> ::= {CREate APPend}
:SYSTem:SETup <setup_data> (see page 1457)	:SYSTem:SETup? (see page 1457)	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:SHUTdown (see page 1459)	n/a	n/a
:SYSTem:TIME <time> (see page 1460)	:SYSTem:TIME? (see page 1460)	<time> ::= hours, minutes, seconds in NR1 format
n/a	:SYSTem:TIME:DSAVing? (see page 1461)	<setting> ::= {0 1}
:SYSTem:TIME:NTPRotoc ol {{0 OFF} {1 ON}} (see page 1462)	:SYSTem:TIME:NTPRotoc ol? (see page 1462)	<setting> ::= {0 1}

Table 69 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:TIME:ZONE <zone> (see page 1463)	:SYSTem:TIME:ZONE? (see page 1463)	<zone> ::= {UP0000 Europe_London UP0100 Europe_Paris UP0200 Africa_Cairo UP0300 UP0330 Asia_Tehran UP0400 Asia_Dubai UP0430 Asia_Kabul UP0500 Asia_Karachi UP0530 Asia_Kolkata UP0600 Asia_Dhaka UP0700 Asia_Bangkok UP0800 Asia_Hong_Kong UP0845 Australia_Eucla UP0900 Asia_Tokyo UP0930 Australia_Adelaide Australia_Darwin UP1000 Australia_Brisbane Australia_Sydney UP1030 Australia_Lord_Howe UP1100 Pacific_Noumea UP1200 UM0100 UM0200 UM0300 America_Sao_Paulo UM0330 America_St_Johns UM0400 America_Caracas UM0500 America_New_York UM0600 America_Chicago UM0700 America_Denver UM0800 America_Los_Angeles UM0900 America_Anchorage UM1000 Pacific_Honolulu UM1100 UM1200}
:SYSTem:TOUCh {{1 ON} {0 OFF}} (see page 1464)	:SYSTem:TOUCh? (see page 1464)	{1 0}
n/a	:SYSTem:VERSion? (see page 1465)	<SCPI_version_implemented> ::= 1999.0

Table 70 :TIMebase Commands Summary

Command	Query	Options and Query Returns
:TIMebase[:MAIN]:POSi tion <pos> (see page 1469)	:TIMebase[:MAIN]:POSi tion? (see page 1469)	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMebase[:MAIN]:RANG e <range_value> (see page 1470)	:TIMebase[:MAIN]:RANG e? (see page 1470)	<range_value> ::= time for 10 div in seconds in NR3 format

Table 70 :TImebase Commands Summary (continued)

Command	Query	Options and Query Returns
:TImebase[:MAIN]:SCALe <scale_value> (see page 1471)	:TImebase[:MAIN]:SCALe? (see page 1471)	<scale_value> ::= time/div in seconds in NR3 format
:TImebase:MODE <value> (see page 1472)	:TImebase:MODE? (see page 1472)	<value> ::= {MAIN WINDow XY ROLL}
:TImebase:REfERENCE {LEFT CENTer RIGHT CUSTom} (see page 1473)	:TImebase:REfERENCE? (see page 1473)	<return_value> ::= {LEFT CENTer RIGHT CUSTom}
:TImebase:REfERENCE:L OCation <loc> (see page 1474)	:TImebase:REfERENCE:L OCation? (see page 1474)	<loc> ::= 0.0 to 1.0 in NR3 format
:TImebase:VERNIer {{0 OFF} {1 ON}} (see page 1475)	:TImebase:VERNIer? (see page 1475)	{0 1}
:TImebase:WINDow:POSi tion <pos> (see page 1476)	:TImebase:WINDow:POSi tion? (see page 1476)	<pos> ::= time from the trigger event to the zoomed view reference point in NR3 format
:TImebase:WINDow:RANGe <range_value> (see page 1477)	:TImebase:WINDow:RANGe? (see page 1477)	<range value> ::= range value in seconds in NR3 format for the zoomed window
:TImebase:WINDow:SCALe <scale_value> (see page 1478)	:TImebase:WINDow:SCALe? (see page 1478)	<scale_value> ::= scale value in seconds in NR3 format for the zoomed window

Table 71 General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:FORCe (see page 1483)	n/a	n/a
:TRIGger:HFReject {{0 OFF} {1 ON}} (see page 1484)	:TRIGger:HFReject? (see page 1484)	{0 1}
:TRIGger:HOLDOff <holdoff_time> (see page 1485)	:TRIGger:HOLDOff? (see page 1485)	<holdoff_time> ::= 60 ns to 10 s in NR3 format
:TRIGger:HOLDOff:MAXi mum <max_holdoff> (see page 1486)	:TRIGger:HOLDOff:MAXi mum? (see page 1486)	<max_holdoff> ::= maximum holdoff time in seconds in NR3 format

Table 71 General :TRIGger Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:HOLDOff:MINimum <min_holdoff> (see page 1487)	:TRIGger:HOLDOff:MINimum? (see page 1487)	<min_holdoff> ::= minimum holdoff time in seconds in NR3 format
:TRIGger:HOLDOff:RANDOM {{0 OFF} {1 ON}} (see page 1488)	:TRIGger:HOLDOff:RANDOM? (see page 1488)	<setting> ::= {0 1}
:TRIGger:HYSTeresis:STATe {{0 OFF} {1 ON}} (see page 1490)	:TRIGger:HYSTeresis:STATe? (see page 1490)	{0 1}
:TRIGger:JFRee {{0 OFF} {1 ON}} (see page 1489)	:TRIGger:JFRee? (see page 1489)	{0 1}
:TRIGger:LEVel:ASETup (see page 1491)	n/a	n/a
:TRIGger:LEVel:HIGH <level>, <source> (see page 1492)	:TRIGger:LEVel:HIGH? <source> (see page 1492)	<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 1493)	:TRIGger:LEVel:LOW? <source> (see page 1493)	<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 1494)	:TRIGger:MODE? (see page 1494)	<mode> ::= {EDGE GLITCh PATtern DELay OR RUNT SHOLd TRANSition {PBUS{1 2} SBUS{1 2}} USB} <return_value> ::= {<mode> <none>} <none> ::= query returns "NONE" if the :TIMEbase:MODE is ROLL or XY
:TRIGger:NREJect {{0 OFF} {1 ON}} (see page 1495)	:TRIGger:NREJect? (see page 1495)	{0 1}
:TRIGger:SWEep <sweep> (see page 1496)	:TRIGger:SWEep? (see page 1496)	<sweep> ::= {AUTO NORMal}

Table 72 :TRIGger:DElay Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DElay:ARM:SL OPe <slope> (see page 1499)	:TRIGger:DElay:ARM:SL OPe? (see page 1499)	<slope> ::= {NEGative POSitive}
:TRIGger:DElay:ARM:SO URce <source> (see page 1500)	:TRIGger:DElay:ARM:SO URce? (see page 1500)	<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 :TMAX <time_value> (see page 1501)	:TRIGger:DElay:TDElay :TMAX? (see page 1501)	<time_value> ::= time in seconds in NR3 format
:TRIGger:DElay:TDElay :TMIN <time_value> (see page 1502)	:TRIGger:DElay:TDElay :TMIN? (see page 1502)	<time_value> ::= time in seconds in NR3 format
:TRIGger:DElay:TRIGge r:SLOPe <slope> (see page 1503)	:TRIGger:DElay:TRIGge r:SLOPe? (see page 1503)	<slope> ::= {NEGative POSitive}
:TRIGger:DElay:TRIGge r:SOURce <source> (see page 1504)	:TRIGger:DElay:TRIGge r:SOURce? (see page 1504)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 73 :TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:COUPling {AC DC LFReject} (see page 1506)	:TRIGger[:EDGE]:COUPling? (see page 1506)	{AC DC LFReject}
:TRIGger[:EDGE]:LEVel <level> [, <source>] (see page 1507)	:TRIGger[:EDGE]:LEVel? [<source>] (see page 1507)	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, <level> ::= ±8 V. <source> ::= {CHANnel<n> DIGital<d> EXTernal } <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 1508)	:TRIGger[:EDGE]:REJect? (see page 1508)	{OFF LFReject HFReject}
:TRIGger[:EDGE]:SLOPe <polarity> (see page 1509)	:TRIGger[:EDGE]:SLOPe? (see page 1509)	<polarity> ::= {POSitive NEGative EITHER ALTernate}
:TRIGger[:EDGE]:SOURC e <source> (see page 1510)	:TRIGger[:EDGE]:SOURC e? (see page 1510)	<source> ::= {CHANnel<n> DIGital<d> EXTernal LINE WGEN} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 74 :TRIGger:GLITch Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITch:GREAt erthan <greater_than_time>[s uffix] (see page 1513)	:TRIGger:GLITch:GREAt erthan? (see page 1513)	<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 page 1514)	:TRIGger:GLITch:LESSt han? (see page 1514)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:GLITch:LEVel <level> [<source>] (see page 1515)	:TRIGger:GLITch:LEVel ? (see page 1515)	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For digital channels, <level> ::= ± 8 V. <source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:GLITch:POLar ity <polarity> (see page 1516)	:TRIGger:GLITch:POLar ity? (see page 1516)	<polarity> ::= {POSitive NEGative}
:TRIGger:GLITch:QUALi fier <qualifier> (see page 1517)	:TRIGger:GLITch:QUALi fier? (see page 1517)	<qualifier> ::= {GREaterthan LESSthan RANGE}
:TRIGger:GLITch:RANGe <less_than_time>[suff ix], <greater_than_time>[s uffix] (see page 1518)	:TRIGger:GLITch:RANGe ? (see page 1518)	<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:SOURc e <source> (see page 1519)	:TRIGger:GLITch:SOURc e? (see page 1519)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 75 :TRIGger:OR Commands Summary

Command	Query	Options and Query Returns
:TRIGger:OR <string> (see page 1521)	:TRIGger:OR? (see page 1521)	<string> ::= "nn...n" where n ::= {R F E X} R = rising edge, F = falling edge, E = either edge, X = don't care. Each character in the string is for an analog or digital channel as shown on the front panel display.

Table 76 :TRIGger:PATtern Commands Summary

Command	Query	Options and Query Returns
:TRIGger:PATtern <string>[,<edge_source>,<edge>] (see page 1523)	:TRIGger:PATtern? (see page 1524)	<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> DIGital<d> NONE} <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 1525)	:TRIGger:PATtern:FORM at? (see page 1525)	<base> ::= {ASCII HEX}
:TRIGger:PATtern:GREa terthan <greater_than_time>[suffix] (see page 1526)	:TRIGger:PATtern:GREa terthan? (see page 1526)	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:PATtern:LESS than <less_than_time>[suffix] (see page 1527)	:TRIGger:PATtern:LESS than? (see page 1527)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 76 :TRIGger:PATtern Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:PATtern:QUALifier <qualifier> (see page 1528)	:TRIGger:PATtern:QUALifier? (see page 1528)	<qualifier> ::= {ENTERed GREATERthan LESSthan INRange OUTRange TIMEout}
:TRIGger:PATtern:RANGe <less_than_time>[suffix], <greater_than_time>[suffix] (see page 1529)	:TRIGger:PATtern:RANGe? (see page 1529)	<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 77 :TRIGger:RUNT Commands Summary

Command	Query	Options and Query Returns
:TRIGger:RUNT:POLarity <polarity> (see page 1531)	:TRIGger:RUNT:POLarity? (see page 1531)	<polarity> ::= {POSitive NEGative}
:TRIGger:RUNT:QUALifier <qualifier> (see page 1532)	:TRIGger:RUNT:QUALifier? (see page 1532)	<qualifier> ::= {GREATERthan LESSthan NONE}
:TRIGger:RUNT:SOURce <source> (see page 1533)	:TRIGger:RUNT:SOURce? (see page 1533)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:RUNT:TIME <time>[suffix] (see page 1534)	:TRIGger:RUNT:TIME? (see page 1534)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 78 :TRIGger:SHOLd Commands Summary

Command	Query	Options and Query Returns
:TRIGger:SHOLd:SLOPe <slope> (see page 1536)	:TRIGger:SHOLd:SLOPe? (see page 1536)	<slope> ::= {NEGative POSitive}
:TRIGger:SHOLd:SOURce:CLOCK <source> (see page 1537)	:TRIGger:SHOLd:SOURce:CLOCK? (see page 1537)	<source> ::= {CHANNEL<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 78 :TRIGger:SHOLd Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:SHOLd:SOURce :DATA <source> (see page 1538)	:TRIGger:SHOLd:SOURce :DATA? (see page 1538)	<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 1539)	:TRIGger:SHOLd:TIME:H OLD? (see page 1539)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:SHOLd:TIME:S ETup <time>[suffix] (see page 1540)	:TRIGger:SHOLd:TIME:S ETup? (see page 1540)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 79 :TRIGger:TRANSition Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TRANSition:Q UALifier <qualifier> (see page 1542)	:TRIGger:TRANSition:Q UALifier? (see page 1542)	<qualifier> ::= {GREaterthan LESSthan}
:TRIGger:TRANSition:S LOPe <slope> (see page 1543)	:TRIGger:TRANSition:S LOPe? (see page 1543)	<slope> ::= {NEGative POSitive}
:TRIGger:TRANSition:S OURce <source> (see page 1544)	:TRIGger:TRANSition:S OURce? (see page 1544)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TRANSition:T IME <time>[suffix] (see page 1545)	:TRIGger:TRANSition:T IME? (see page 1545)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

Table 80 :TRIGger:ZONE Commands Summary

Command	Query	Options and Query Returns
:TRIGger:ZONE<z>:LOGic {AND OR} (see page 1547)	:TRIGger:ZONE<z>:LOGic? (see page 1551)	{AND OR} <z> ::= 2-4 in NR1 format
:TRIGger:ZONE<z>:MODE <mode> (see page 1548)	:TRIGger:ZONE<z>:MODE? (see page 1548)	<mode> ::= {INTERsect NOTintersect} <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:PLACement <width>, <height>, <x_center>, <y_center> (see page 1549)	:TRIGger:ZONE<z>:PLACement? (see page 1549)	<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 <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:SOURce <source> (see page 1550)	:TRIGger:ZONE<z>:SOURce? (see page 1550)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:STATe {{0 OFF} {1 ON}} (see page 1551)	:TRIGger:ZONE<z>:STATe? (see page 1551)	{0 1} <z> ::= 1-4 in NR1 format
n/a	:TRIGger:ZONE<z>:VALidity? (see page 1552)	<value> ::= {VALid INValid OSCreen} <z> ::= 1-4 in NR1 format

Table 81 :WAVEform Commands Summary

Command	Query	Options and Query Returns
:WAVEform:BYTeorder <value> (see page 1561)	:WAVEform:BYTeorder? (see page 1561)	<value> ::= {LSBFirst MSBFirst}
n/a	:WAVEform:COUNt? (see page 1562)	<count> ::= an integer from 1 to 65536 in NR1 format

Table 81 :WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVEform:DATA? (see page 1563)	<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 1565)	:WAVEform:FORMat? (see page 1565)	<value> ::= {WORD BYTE ASCII}
:WAVEform:POINTs <wfm_points> (see page 1566)	:WAVEform:POINTs? (see page 1566)	<wfm_points> ::= <#_points> <#_points> ::= an integer in NR1 format
:WAVEform:POINTs:MODE <points_mode> (see page 1568)	:WAVEform:POINTs:MODE ? (see page 1569)	<points_mode> ::= {NORMAL MAXimum RAW}
n/a	:WAVEform:PREamble? (see page 1570)	<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: • 0 for BYTE format • 1 for WORD format • 2 for ASCii format <type> ::= an integer in NR1 format: • 0 for NORMAl type • 1 for PEAK detect type • 3 for AVERAge type <count> ::= Average count, or 1 if PEAK detect type or NORMAl; an integer in NR1 format

Table 81 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVeform:SEGmented:ALL {{0 OFF} {1 ON}} (see page 1573)	:WAVeform:SEGmented:ALL? (see page 1573)	<setting> ::= {0 1}
n/a	:WAVeform:SEGmented:COUNt? (see page 1574)	<count> ::= an integer from 2 to 1000 in NR1 format
n/a	:WAVeform:SEGmented:TAG? (see page 1575)	<time_tag> ::= in NR3 format
n/a	:WAVeform:SEGmented:XLIST? <xlist_type> (see page 1576)	<xlist_type> ::= {RELXorigin ABSXorigin TTAG} <return_value> ::= X-info for all segments
:WAVeform:SOURce <source> (see page 1577)	:WAVeform:SOURce? (see page 1577)	<source> ::= {CHANnel<n> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} SBUS{1 2}}} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:WAVeform:SOURce:SUBSource <subsource> (see page 1581)	:WAVeform:SOURce:SUBSource? (see page 1581)	<subsource> ::= {{SUB0 RX MOSI FAST} {SUB1 TX MISO SLOW}}
n/a	:WAVeform:TYPE? (see page 1582)	<return_mode> ::= {NORM PEAK AVER}
:WAVeform:UNSigned {{0 OFF} {1 ON}} (see page 1583)	:WAVeform:UNSigned? (see page 1583)	{0 1}
:WAVeform:VIEW <view> (see page 1584)	:WAVeform:VIEW? (see page 1584)	<view> ::= {MAIN ALL}
n/a	:WAVeform:XINCrement? (see page 1585)	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVeform:XORigin? (see page 1586)	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVeform:XREFerence? (see page 1587)	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)

Table 81 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVeform:YINCrement? (see page 1588)	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVeform:YORigin? (see page 1589)	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVeform:YREFerence? (see page 1590)	<return_value> ::= y-reference value in the current preamble in NR1 format

Table 82 :WGEN<w> Commands Summary

Command	Query	Options and Query Returns
:WGEN<w>:ARBitrary:BYTeorder <order> (see page 1596)	:WGEN<w>:ARBitrary:BYTeorder? (see page 1596)	<order> ::= {MSBFirst LSBFirst} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARBitrary:DATA {<binary> <value>, <value> ...} (see page 1597)	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 to (# WaveGen outputs) in NR1 format
n/a	:WGEN<w>:ARBitrary:DATA:ATTRibute:POINTs? (see page 1600)	<points> ::= number of points in NR1 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARBitrary:DATA:CLEar (see page 1601)	n/a	<w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARBitrary:DATA:DAC {<binary> <value>, <value> ...} (see page 1602)	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 to (# WaveGen outputs) in NR1 format

Table 82 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:ARBitrary:IN Terpolate {{0 OFF} {1 ON}} (see page 1603)	:WGEN<w>:ARBitrary:IN Terpolate? (see page 1603)	{0 1} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARBitrary:ST ORe <source> (see page 1604)	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 to (# WaveGen outputs) in NR1 format
:WGEN<w>:DCMode <mode> (see page 1605)	:WGEN<w>:DCMode? (see page 1605)	<mode> ::= {PREcise WIDerange}
:WGEN<w>:FREQuency <frequency> (see page 1606)	:WGEN<w>:FREQuency? (see page 1606)	<frequency> ::= frequency in Hz in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:FUNctIon <signal> (see page 1607)	:WGEN<w>:FUNctIon? (see page 1612)	<signal> ::= {SINusoid SQUare RAMP PULSe NOISe DC SINC EXPRise EXPFall CARDiac GAUSSian ARBitrary} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:FUNctIon:PUL Se:WIDTh <width> (see page 1613)	:WGEN<w>:FUNctIon:PUL Se:WIDTh? (see page 1613)	<width> ::= pulse width in seconds in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:FUNctIon:SQU are:DCYClE <percent> (see page 1615)	:WGEN<w>:FUNctIon:SQU are:DCYClE? (see page 1615)	<percent> ::= duty cycle percentage in NR1 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:MODulation:A M:DEPTh <percent> (see page 1616)	:WGEN<w>:MODulation:A M:DEPTh? (see page 1616)	<percent> ::= AM depth percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format

Table 82 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:MODulation:AM:FREQuency <frequency> (see page 1617)	:WGEN<w>:MODulation:AM:FREQuency? (see page 1617)	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FM:DEVIation <frequency> (see page 1618)	:WGEN<w>:MODulation:FM:DEVIation? (see page 1618)	<frequency> ::= frequency deviation in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FM:FREQuency <frequency> (see page 1619)	:WGEN<w>:MODulation:FM:FREQuency? (see page 1619)	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:FUNCTION <shape> (see page 1620)	:WGEN<w>:MODulation:FUNCTION? (see page 1620)	<shape> ::= {SINusoid} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:NOISE <percent> (see page 1621)	:WGEN<w>:MODulation:NOISE? (see page 1621)	<percent> ::= 0 to 100 <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:STATe {{0 OFF} {1 ON}} (see page 1622)	:WGEN<w>:MODulation:STATe? (see page 1622)	{0 1} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:TYPE <type> (see page 1623)	:WGEN<w>:MODulation:TYPE? (see page 1623)	<type> ::= {AM FM} <w> ::= 1 in NR1 format
:WGEN<w>:OUTPut {{0 OFF} {1 ON}} (see page 1624)	:WGEN<w>:OUTPut? (see page 1624)	{0 1} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:OUTPut:BURSt (see page 1625)	n/a	n/a
:WGEN<w>:OUTPut:BURSt:COUNT <number> (see page 1626)	:WGEN<w>:OUTPut:BURSt:COUNT? (see page 1626)	<w> ::= 1 to (# WaveGen outputs) in NR1 format <number> ::= 1 to 10000 in NR1 format
:WGEN<w>:OUTPut:LOAD <impedance> (see page 1627)	:WGEN<w>:OUTPut:LOAD? (see page 1627)	<impedance> ::= {ONEMeg FIFTy} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:OUTPut:MODE <mode> (see page 1628)	:WGEN<w>:OUTPut:MODE? (see page 1628)	<mode> ::= {NORMal BURSt}

Table 82 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:OUTPut:POLarity <polarity> (see page 1629)	:WGEN<w>:OUTPut:POLarity? (see page 1629)	<polarity> ::= {NORMAL INVERTed} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:PERiod <period> (see page 1630)	:WGEN<w>:PERiod? (see page 1630)	<period> ::= period in seconds in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:RST (see page 1631)	n/a	<w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage <amplitude> (see page 1632)	:WGEN<w>:VOLTage? (see page 1632)	<amplitude> ::= amplitude in volts in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:HIGH <high> (see page 1633)	:WGEN<w>:VOLTage:HIGH? (see page 1633)	<high> ::= high-level voltage in volts, in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:LOW <low> (see page 1634)	:WGEN<w>:VOLTage:LOW? (see page 1634)	<low> ::= low-level voltage in volts, in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:OFFSet <offset> (see page 1635)	:WGEN<w>:VOLTage:OFFSet? (see page 1635)	<offset> ::= offset in volts in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format

Table 83 :WMEMemory<r> Commands Summary

Command	Query	Options and Query Returns
:WMEMemory<r>:CLEar (see page 1639)	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format
:WMEMemory<r>:DISPlay {{0 OFF} {1 ON}} (see page 1640)	:WMEMemory<r>:DISPlay? (see page 1640)	<r> ::= 1 to (# ref waveforms) in NR1 format {0 1}

Table 83 :WMEemory<r> Commands Summary (continued)

Command	Query	Options and Query Returns
:WMEemory<r>:LAbel <string> (see page 1641)	:WMEemory<r>:LAbel? (see page 1641)	<r> ::= 1 to (# ref waveforms) in NR1 format <string> ::= any series of 32 or less ASCII characters enclosed in quotation marks
:WMEemory<r>:SAVE <source> (see page 1642)	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: Math functions whose x-axis is not frequency can be saved as reference waveforms.
:WMEemory<r>:SKEW <skew> (see page 1643)	:WMEemory<r>:SKEW? (see page 1643)	<r> ::= 1 to (# ref waveforms) in NR1 format <skew> ::= time in seconds in NR3 format
:WMEemory<r>:YOFFset <offset>[suffix] (see page 1644)	:WMEemory<r>:YOFFset? (see page 1644)	<r> ::= 1 to (# ref waveforms) in NR1 format <offset> ::= vertical offset value in NR3 format [suffix] ::= {V mV}
:WMEemory<r>:YRANge <range> (see page 1645)	:WMEemory<r>:YRANge? (see page 1645)	<r> ::= 1 to (# ref waveforms) in NR1 format <range> ::= vertical full-scale range value in NR3 format
:WMEemory<r>:YSCale <scale> (see page 1646)	:WMEemory<r>:YSCale? (see page 1646)	<r> ::= 1 to (# ref waveforms) in NR1 format <scale> ::= vertical units per division value in NR3 format

Syntax Elements

- **"Number Format"** on page 227
- **"<NL> (Line Terminator)"** on page 227
- **"[] (Optional Syntax Terms)"** on page 227
- **"{ } (Braces)"** on page 227
- **"::= (Defined As)"** on page 227
- **"< > (Angle Brackets)"** on page 228
- **"... (Ellipsis)"** on page 228
- **"n,...,p (Value Ranges)"** on page 228
- **"d (Digits)"** on page 228
- **"Quoted ASCII String"** on page 228
- **"Definite-Length Block Response Data"** on page 228

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> ::= indicates that <A> can be replaced by 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

6 Common (*) Commands

Commands defined by IEEE 488.2 standard that are common to all instruments.
See "[Introduction to Common \(*\) Commands](#)" on page 235.

Table 84 Common (*) Commands Summary

Command	Query	Options and Query Returns
*CLS (see page 236)	n/a	n/a
*ESE <mask> (see page 237)	*ESE? (see page 238)	<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 page 239)	<status> ::= 0 to 255; an integer in NR1 format
n/a	*IDN? (see page 239)	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 page 242)	<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format

Table 84 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
*OPC (see page 243)	*OPC? (see page 243)	ASCII "1" is placed in the output queue when all pending device operations have completed.
n/a	*OPT? (see page 244)	<pre> <return_value> ::= 0,0,<license info> <license info> ::= <All field>, <reserved>, <Enhanced Security>, <Waveform Generator>, <Automotive Serial>, <Aerospace Serial>, <Embedded Serial>, <USB Serial>, <Power Measurements>, <Jitter Analysis>, <reserved>, <reserved>, <reserved>, <Memory 100M>, <Memory 50M>, <Memory 20M>, <Bandwidth 1GHz>, <Bandwidth 500MHz>, <Bandwidth 350MHz>, <Bandwidth 200MHz>, <Demo Partner>, <Demo Keysight> </pre>

Table 84 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	*OPT? (see page 244) (cont'd)	<All field> ::= {0 All(d)} <reserved> ::= 0 <Enhanced Security> ::= {0 HD3SECURE} <Waveform Generator> ::= {0 WAVEGEN} <Automotive Serial> ::= {0 HD300AUTA} <Aerospace Serial> ::= {0 HD300AERA} <Embedded Serial> ::= {0 HD300EMBA} <USB Serial> ::= {0 HD300USBA} <Power Measurements> ::= {0 HD300PWRA} <Jitter Analysis> ::= {0 HD300QJTA} <Memory 100M> ::= {0 MEMUP100} <Memory 50M> ::= {0 MEMUP50} <Memory 20M> ::= {0 MEMUP20} <Bandwidth 1GHz> ::= {0 BW1000} <Bandwidth 500MHz> ::= {0 BW500} <Bandwidth 350MHz> ::= {0 BW350} <Bandwidth 200MHz> ::= {0 BW200} <Demo Partner> ::= {0 DIS} <Demo Keysight> ::= {0 D24}
*RCL {<internal_fname> <external_fname>} (see page 246)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
*RST (see page 247)	n/a	See *RST (Reset) (see page 247)

Table 84 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns
*SAV {<internal_fname> <external_fname>} (see page 250)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
*SRE <mask> (see page 251)	*SRE? (see page 252)	<mask> ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. <mask> ::= following values: 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 253)	<value> ::= 0 to 255; an integer in NR1 format, as shown in the following: 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 256)	n/a	n/a
n/a	*TST? (see page 257)	<result> ::= 0 or non-zero value; an integer in NR1 format
*WAI (see page 258)	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)

C (see [page 1708](#))

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 235
 - ["*STB? \(Read Status Byte\)"](#) on page 253
 - ["*ESE \(Standard Event Status Enable\)"](#) on page 237
 - ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - ["*SRE \(Service Request Enable\)"](#) on page 251
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432
 - [":STATus:PRESet"](#) on page 1287

*ESE (Standard Event Status Enable)

C (see [page 1708](#))

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.

**Standard Event
Status Enable
Register**

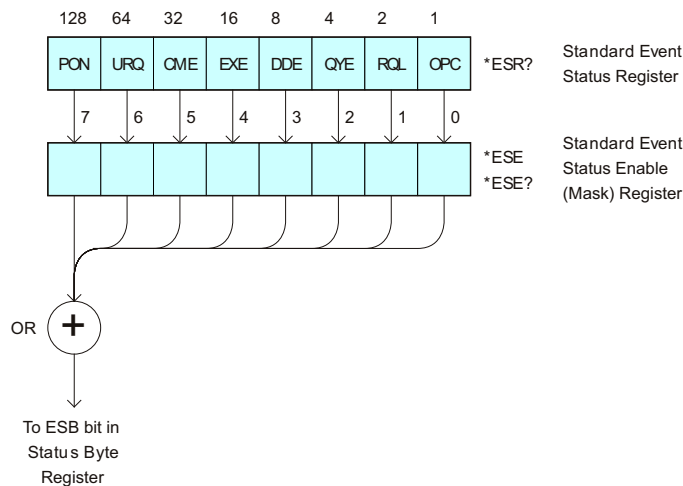


Table 85 Standard Event Status Enable (ESE) Register Bits

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.

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.

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 235
 - ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - ["*OPC \(Operation Complete\)"](#) on page 243
 - ["*CLS \(Clear Status\)"](#) on page 236

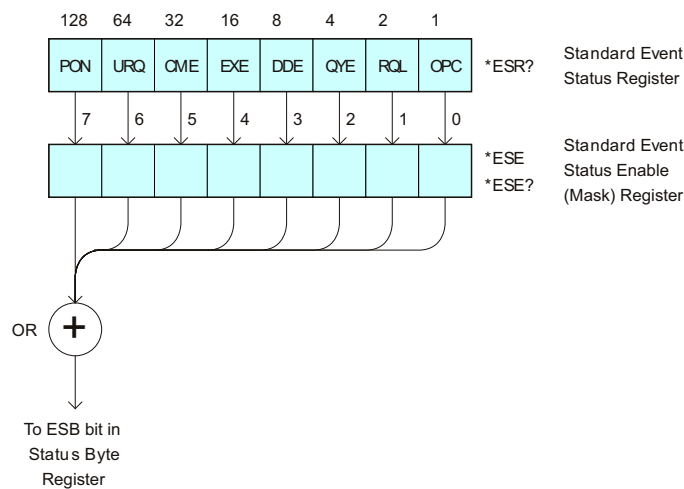
*ESR? (Standard Event Status Register)

C (see [page 1708](#))

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.

Standard Event Status Register



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

Table 86 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.

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.

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.

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 235
 - ["*ESE \(Standard Event Status Enable\)"](#) on page 237
 - ["*OPC \(Operation Complete\)"](#) on page 243
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432

*IDN? (Identification Number)

C (see [page 1708](#))

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 235
- ["*OPT? \(Option Identification\)"](#) on page 244
- [":SYSTem:PERSonA\[:MANUFACTURER\]"](#) on page 1440
- [":SYSTem:PERSonA\[:MANUFACTURER\]:DEFAULT"](#) on page 1441

*LRN? (Learn Device Setup)

C (see [page 1708](#))

Query Syntax

*LRN?

The *LRN? query result contains the current state of the instrument. This query is similar to the :SYSTem:SETup? (see [page 1457](#)) 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 235
 - ["*RCL \(Recall\)"](#) on page 246
 - ["*SAV \(Save\)"](#) on page 250
 - [":SYSTem:SETup"](#) on page 1457

*OPC (Operation Complete)

C (see [page 1708](#))

Command Syntax `*OPC`

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

You can use the *ESR? query to look at the OPC bit (bit 0) in the Standard Event Status Register to determine when an operation is complete.

NOTE

The front-panel graphical user interface can disable most of the remote interface, including the *OPC syntax, in certain situations. Bit 4 in the Operation Status Register shows whether the remote user interface is enabled or disabled. For more information, see [":STATus:OPERation:CONDition?"](#) on page 1298.

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.

The *OPC? query can be used between overlapped commands to make sure one is complete before the next one begins. The *OPC? query can also be used to pause a controller program until an operation is complete.

Return Format `<complete><NL>`

`<complete> ::= 1`

- See Also
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*ESE \(Standard Event Status Enable\)"](#) on page 237
 - ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [Chapter 4](#), "Sequential (Blocking) vs. Overlapped Commands," starting on page 91
 - ["Synchronization with an Averaging Acquisition"](#) on page 1693
 - ["Set Up the Oscilloscope"](#) on page 1686
 - ["Example: Blocking and Polling Synchronization"](#) on page 1695
 - ["Example: Waiting for IO Operation Complete"](#) on page 1682

*OPT? (Option Identification)

C (see [page 1708](#))

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>, <Enhanced Security>,
<Waveform Generator>, <Automotive Serial>, <Aerospace Serial>,
<Embedded Serial>, <USB Serial>, <Power Measurements>,
<Jitter Analysis>, <reserved>, <reserved>, <reserved>,
<Memory 100M>, <Memory 50M>, <Memory 20M>, <Bandwidth 1GHz>,
<Bandwidth 500MHz>, <Bandwidth 350MHz>, <Bandwidth 200MHz>,
<Demo Partner>, <Demo Keysight>
```

```
<All field> ::= {0 | All(d)}
```

```
<reserved> ::= 0
```

```
<Enhanced Security> ::= {0 | HD3SECURE}
```

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

```
<Automotive Serial> ::= {0 | HD300AUTA}
```

```
<Aerospace Serial> ::= {0 | HD300AERA}
```

```
<Embedded Serial> ::= {0 | HD300EMBA}
```

```
<USB Serial> ::= {0 | HD300USBA}
```

```
<Power Measurements> ::= {0 | HD300PWRA}
```

```
<Jitter Analysis> ::= {0 | HD300QJTA}
```

```
<Memory 100M> ::= {0 | MEMUP100}
```

```
<Memory 50M> ::= {0 | MEMUP50}
```

```
<Memory 20M> ::= {0 | MEMUP20}
```

```
<Bandwidth 1GHz> ::= {0 | BW1000}
```

```
<Bandwidth 500MHz> ::= {0 | BW500}
```

```
<Bandwidth 350MHz> ::= {0 | BW350}
```

```
<Bandwidth 200MHz> ::= {0 | BW200}
```

```
<Demo Partner> ::= {0 | DIS}
```

```
<Demo Keysight> ::= {0 | D24}
```

The *OPT? query returns the following:

Module	Module Id
No modules attached	0,0,All(d),0,HD3SECURE(d),WAVEGEN,HD300AUTA,HD300AERA,HD300EMBA,HD300USBA,HD300PWRA,HD300QJTA,0,0,0,MEMUP100,MEMUP50,MEMUP20,BW1000,BW500,BW350,BW200,DIS,D24

- See Also
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*IDN? \(Identification Number\)"](#) on page 241

*RCL (Recall)

C (see [page 1708](#))

Command Syntax *RCL {<internal_fname> | <external_fname>}

<internal_fname> ::= quoted ASCII file name string beginning
with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning
with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending
on the number of USB storage devices connected

The *RCL command recalls an oscilloscope setup from a file.

- See Also**
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*SAV \(Save\)"](#) on page 250
 - [":RECall:SETup\[:START\]"](#) on page 950

*RST (Reset)

C (see [page 1708](#))

Command Syntax *RST

The *RST command places the instrument in a known state. This is the same as selecting **Factory Default** in the graphical user interface's Default dialog box (**Control > Default...**).

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 Settings	
Mode	Normal
Averaging	Off
# Averages	8

Analog Channel Settings	
Channel 1	On
Channel 2/3/4	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 Settings	
Source	Channel 1

Digital Channel Settings	
Channel 0 - 15	Off
Labels	Off
Threshold	TTL (1.4 V)

Display Settings	
Persistence	Off
Grid	20%

Measurement Settings	
Source	Channel 1

Run Control Settings	
	Scope is running

Time Base Settings	
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

Trigger Settings	
Type	Edge
Mode	Auto
Coupling	dc

Trigger Settings	
Source	Channel 1
Level	0.0 V
Slope	Positive
HF Reject and noise reject	Off
Holdoff	40 ns
External probe attenuation	1:1
External Units	Volts
External Impedance	1 M Ohm (cannot be changed)

- See Also
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - [":SYSTem:PRESet"](#) on page 1445

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 49](#), "Programming Examples," starting on page 1717

*SAV (Save)

C (see [page 1708](#))

Command Syntax *SAV {<internal_fname> | <external_fname>}

<internal_fname> ::= quoted ASCII file name string beginning
with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning
with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending
on the number of USB storage devices connected

The *SAV command saves the oscilloscope setup to a file.

- See Also**
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*RCL \(Recall\)"](#) on page 246
 - [":SAVE:SETup\[:START\]"](#) on page 974

*SRE (Service Request Enable)

C (see [page 1708](#))

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.

Service Request Enable Register

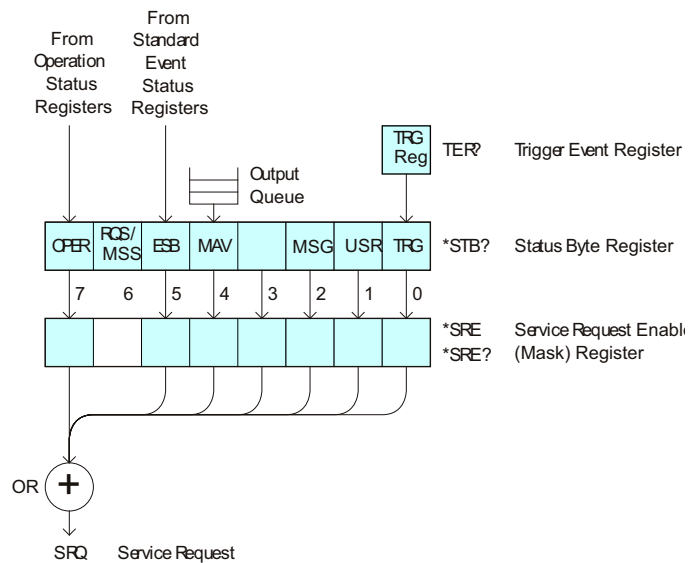


Table 87 Service Request Enable Register (SRE) Bits

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.

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).

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 235
 - ["*STB? \(Read Status Byte\)"](#) on page 253
 - ["*CLS \(Clear Status\)"](#) on page 236

*STB? (Read Status Byte)

C (see [page 1708](#))

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.

Status Byte Register

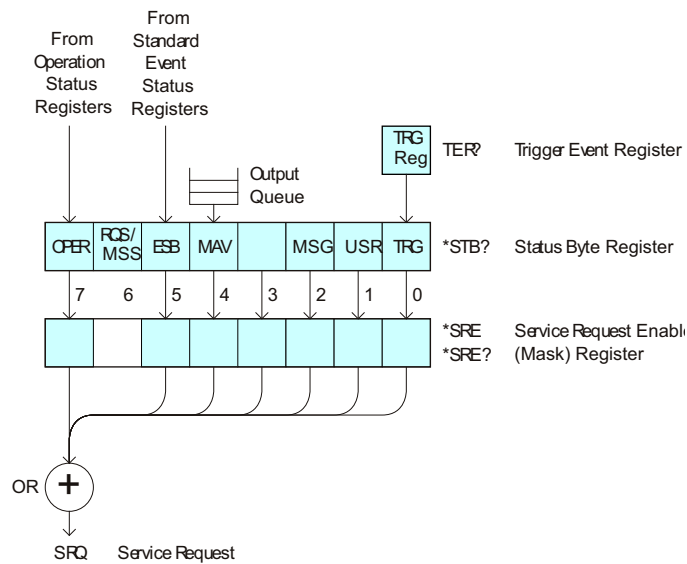


Table 88 Status Byte Register (STB) Bits

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.

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.

Return Format <value><NL>

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

- See Also**
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*SRE \(Service Request Enable\)"](#) on page 251

*TRG (Trigger)

C (see [page 1708](#))

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 235
 - [":DIGitize"](#) on page 271
 - [":RUN"](#) on page 275
 - [":STOP"](#) on page 279

*TST? (Self Test)

C (see [page 1708](#))

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 235

*WAI (Wait To Continue)

C (see [page 1708](#))

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 235

7 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 261.

Table 89 Root (:) Commands Summary

Command	Query	Options and Query Returns
:ACTivity (see page 262)	:ACTivity? (see page 262)	<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 263)	{0 1}; an integer in NR1 format
:AUToscale [<source>[,...,<source>]] (see page 265)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD1 POD2} <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 267)	:AUToscale:AMODE? (see page 267)	<value> ::= {NORMal CURRent}}
:AUToscale:CHANnels <value> (see page 268)	:AUToscale:CHANnels? (see page 268)	<value> ::= {ALL DISPlayed}}
:AUToscale:FDEBug {{0 OFF} {1 ON}} (see page 269)	:AUToscale:FDEBug? (see page 269)	{0 1}

Table 89 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:BLANk [<source>] (see page 270)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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
:DIGitize [<source>[,...,<source>]] (see page 271)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} SBUS{1 2}}} <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 <m> ::= 1 to (# math functions) in NR1 format
n/a	:MODEl? [FRAME] (see page 276)	<return value> ::= string containing the oscilloscope's model number
n/a	:RState? (see page 274)	n/a
:RUN (see page 275)	n/a	n/a
n/a	:SERial? (see page 276)	<return value> ::= quoted string containing serial number
:SINGle (see page 277)	n/a	n/a

Table 89 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:STATus? <display> (see page 278)	{0 1} <display> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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 page 279)	n/a	n/a
n/a	:TER? (see page 280)	{0 1}
:VIEW <source> (see page 281)	n/a	<source> ::= {CHANnel<n> DIGital<d> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} 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

**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 1708](#))

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 261
 - [":POD<n>:THReshold"](#) on page 835
 - [":DIGital<d>:THReshold"](#) on page 404

:AER? (Arm Event Register)

C (see [page 1708](#))

Query Syntax :AER?

The :AER? query reads the Arm Event Register. After the Arm Event Register is read, it is cleared.

A return value of "1" indicates the trigger system has been in the armed state (ready to accept a trigger) since the register was last read.

A return value of "0" indicates the trigger system has not been in the armed state since the register was last read.

If the Arm Event Register is read (and cleared) while the oscilloscope is RUNNING, it will quickly reset to "1" because the oscilloscope either remains in the armed state when triggers are not being found or re-arms quickly after triggers are found and acquired data is saved.

Likewise, if the Arm Event Register is read (and cleared) while the oscilloscope is performing a SINGLE acquisition and the trigger event has not been found, the Arm Event Register will quickly reset to "1" because the oscilloscope remains in the armed state.

NOTE

The HD3-Series oscilloscope's :AER? query behavior is different than in previous InfiniiVision oscilloscopes where the Arm Event Register is set once once per RUN or SINGLE.

The Arm 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).

NOTE

The :AER? query is an alias for the :STATus:OPERation:ARM:BIT0[:EVENT]? and :STATus:OPERation:ARM[:EVENT]? queries.

Return Format <value><NL>

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

- See Also
- ["Introduction to Root \(:\) Commands"](#) on page 261
 - [":STATus:OPERation:ENABLE"](#) on page 1299
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - ["*STB? \(Read Status Byte\)"](#) on page 253

- **"*SRE (Service Request Enable)"** on page 251
- **":STATus:OPERation:ARM:BIT[:EVENT]?"** on page 1310
- **":STATus:OPERation:ARM[:EVENT]?"** on page 1315

:AUToscale

C (see [page 1708](#))

Command Syntax :AUToscale

```
:AUToscale [<source>[,...,<source>]]
```

```
<source> ::= {DIGital<d> | POD1 | POD2 | CHANnel<n>}
```

```
<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 268) 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*.

See Also • ["Introduction to Root \(:\) Commands"](#) on page 261

- **":AUToscale:CHANnels"** on page 268
- **":AUToscale:AMODE"** on page 267

Example Code

```
' 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.
```

See complete example programs at: **Chapter 49**, “Programming Examples,”
starting on page 1717

:AUToscale:AMODE

N (see [page 1708](#))

Command Syntax :AUToscale:AMODE <value>
 <value> ::= {NORMal | CURRent}

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 CURRent 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 261
 - [":AUToscale"](#) on page 265
 - [":AUToscale:CHANnels"](#) on page 268
 - [":ACquire:TYPE"](#) on page 311
 - [":ACquire:MODE"](#) on page 296

:AUToscale:CHANnels

N (see [page 1708](#))

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 261
 - [":AUToscale"](#) on page 265
 - [":AUToscale:AMODe"](#) on page 267
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270

:AUToscale:FDEBug

N (see [page 1708](#))

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 261
- [":AUToscale"](#) on page 265

:BLANK

N (see [page 1708](#))

Command Syntax

:BLANK [<source>]

```
<source> ::= {CHANnel<n> | DIGital<d> | POD{1 | 2} | BUS{1 | 2}
             | FUNCTION<m> | MATH<m> | FFT | {PBUS{1 | 2} | 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

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 261
 - [":DISPlay:CLEAr"](#) on page 420
 - [":CHANnel<n>:DISPlay"](#) on page 351
 - [":DIGital<d>:DISPlay"](#) on page 400
 - [":FUNCTION<m>:DISPlay"](#) on page 533
 - [":POD<n>:DISPlay"](#) on page 831
 - [":SBUS<n>:DISPlay"](#) on page 986
 - [":WMemory<r>:DISPlay"](#) on page 1640
 - [":STATus?"](#) on page 278
 - [":VIEW"](#) on page 281

- Example Code
- ["Example Code"](#) on page 281

:DIGitize

C (see [page 1708](#))

Command Syntax :DIGitize [<source>[,...,<source>]]

<source> ::= {CHANnel<n> | DIGital<d> | POD{1 | 2} | BUS{1 | 2}
| FUNcTion<m> | MATH<m> | FFT | {PBUS{1 | 2} | SBUS{1 | 2}}}

<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 261
 - [":RUN"](#) on page 275
 - [":SINGle"](#) on page 277
 - [":STOP"](#) on page 279
 - [":TIMEbase:MODE"](#) on page 1472
 - [Chapter 8](#), “:ACQuire Commands,” starting on page 283
 - [Chapter 41](#), “:WAVEform Commands,” starting on page 1553
 - [Chapter 4](#), “Sequential (Blocking) vs. Overlapped Commands,” starting on page 91
 - ["Example: Checking for Armed Status"](#) on page 1677

Example Code

```
' Capture an acquisition using :DIGitize.
```

```
' -----  
myScope.WriteString ":DIGitize CHANnel1"
```

See complete example programs at: **Chapter 49**, "Programming Examples," starting on page 1717

:MODEl?

N (see [page 1708](#))

Query Syntax :MODEl? [FRAME]

The :MODEl? query returns the oscilloscope's model number.

The query is the same with or without the FRAME option.

Return Format: string<NL>

See Also • ["Introduction to Root \(:\) Commands"](#) on page 261

:RSTate?

N (see [page 1708](#))

Query Syntax :RSTate?

The :RSTate? query returns the run state:

- RUN – The oscilloscope is acquiring and displaying new waveforms.
- STOP – The oscilloscope is no longer acquiring new waveforms.
- SING – A single acquisition has been started and the oscilloscope is waiting for the trigger condition to be met.

These are the same run states displayed on the front panel and in the user interface.

Return Format {RUN | STOP | SING}<NL>

- See Also**
- [":RUN"](#) on page 275
 - [":SINGle"](#) on page 277
 - [":STOP"](#) on page 279
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302

:RUN

C (see [page 1708](#))

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 261
 - [":SINGle"](#) on page 277
 - [":STOP"](#) on page 279

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 49](#), "Programming Examples," starting on page 1717

:SERial?

N (see [page 1708](#))

Query Syntax :SERial?

The :SERial? query returns the serial number of the instrument.

Return Format: Quoted string<NL>

- See Also**
- ["Introduction to Root \(:\) Commands"](#) on page 261
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:SINGle

C (see [page 1708](#))

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 261
 - [":RUN"](#) on page 275
 - [":STOP"](#) on page 279

:STATus?

N (see [page 1708](#))

Query Syntax :STATus? <source>

```
<source> ::= {CHANnel<n> | DIGital<d> | POD{1 | 2} | BUS{1 | 2}
             | FUNction<m> | MATH<m> | FFT | {PBUS{1 | 2} | 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

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 261
 - [":BLANK"](#) on page 270
 - [":CHANnel<n>:DISPlay"](#) on page 351
 - [":DIGital<d>:DISPlay"](#) on page 400
 - [":FUNction<m>:DISPlay"](#) on page 533
 - [":POD<n>:DISPlay"](#) on page 831
 - [":SBUS<n>:DISPlay"](#) on page 986
 - [":WMEMory<r>:DISPlay"](#) on page 1640
 - [":VIEW"](#) on page 281

:STOP

C (see [page 1708](#))

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 261
 - [":RUN"](#) on page 275
 - [":SINGle"](#) on page 277

- Example Code**
- ["Example Code"](#) on page 275

:TER? (Trigger Event Register)

C (see [page 1708](#))

Query Syntax :TER?

The :TER? query reads the Trigger Event Register. After the Trigger Event Register is read, it is cleared.

A return value of "1" indicates a trigger has occurred since the register was last read.

A return value of "0" indicates a trigger has not occurred since the register was last read.

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.

NOTE

The :TER? query is an alias for the :STATus:TRIGger:BIT0[:EVENT]? and :STATus:TRIGger[:EVENT]? queries.

Return Format <value><NL>

<value> ::= {1 | 0}; a 16-bit integer in NR1 format.

- See Also
- ["Introduction to Root \(:\) Commands"](#) on page 261
 - ["*SRE \(Service Request Enable\)"](#) on page 251
 - ["*STB? \(Read Status Byte\)"](#) on page 253
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406

:VIEW

N (see [page 1708](#))

Command Syntax

```
:VIEW <source>
```

```
<source> ::= {CHANnel<n> | DIGital<d> | POD{1 | 2} | BUS{1 | 2}
             | FUNCTION<m> | MATH<m> | FFT | {PBUS{1 | 2} | 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
```

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 261
 - [":BLANK"](#) on page 270
 - [":CHANnel<n>:DISPlay"](#) on page 351
 - [":DIGital<d>:DISPlay"](#) on page 400
 - [":FUNCTION<m>:DISPlay"](#) on page 533
 - [":POD<n>:DISPlay"](#) on page 831
 - [":SBUS<n>:DISPlay"](#) on page 986
 - [":WMEMory<r>:DISPlay"](#) on page 1640
 - [":STATus?"](#) on page 278

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 49](#), "Programming Examples," starting on page 1717

8 :ACQuire Commands

Set the parameters for acquiring and storing data. See ["Introduction to :ACQuire Commands"](#) on page 284.

Table 90 :ACQuire Commands Summary

Command	Query	Options and Query Returns
n/a	:ACQuire:AALias? (see page 286)	{1 0}
:ACQuire:COMPLete <complete> (see page 287)	:ACQuire:COMPLete? (see page 287)	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNt <count> (see page 288)	:ACQuire:COUNt? (see page 288)	<count> ::= an integer from 2 to 65536 in NR1 format
:ACQuire:DAALias <mode> (see page 289)	:ACQuire:DAALias? (see page 289)	<mode> ::= {DISable AUTO}
:ACQuire:DIGitizer {0 OFF} {1 ON} (see page 290)	:ACQuire:DIGitizer? (see page 290)	{0 1}
:ACQuire:MANual {0 OFF} {1 ON} (see page 292)	:ACQuire:MANual? (see page 292)	{0 1}
n/a	:ACQuire:MANual:POINTs? (see page 294)	<gui_setting> ::= Max Memory Depth setting in NR1 format
n/a	:ACQuire:MANual:SRATe? (see page 295)	<gui_setting> ::= Max Sample Rate setting in NR3 format
:ACQuire:MODE <mode> (see page 296)	:ACQuire:MODE? (see page 296)	<mode> ::= {RTIME ETIME SEGmented}
:ACQuire:POINTs[:ANALog] <points> (see page 297)	:ACQuire:POINTs[:ANALog]? (see page 297)	<points> ::= {AUTO <points_value>} <points_value> ::= desired analog memory depth in integer NR1 format

Table 90 :ACquire Commands Summary (continued)

Command	Query	Options and Query Returns
:ACquire:POINTs[:ANALog]:AUTO {{0 OFF} {1 ON}} (see page 298)	:ACquire:POINTs[:ANALog]:AUTO? (see page 299)	{0 1}
n/a	:ACquire:POINTs:MAX? (see page 301)	<max_points> ::= an integer in NR1 format
:ACquire:SEGmented:ANALyze (see page 302)	n/a	n/a
:ACquire:SEGmented:COUnT <count> (see page 303)	:ACquire:SEGmented:COUnT? (see page 303)	<count> ::= an integer from 2 to 1000 in NR1 format
:ACquire:SEGmented:INDeX <index> (see page 304)	:ACquire:SEGmented:INDeX? (see page 304)	<index> ::= an integer from 1 to 1000 in NR1 format
:ACquire:SRATe[:ANALog] <rate> (see page 307)	:ACquire:SRATe[:ANALog]? (see page 307)	<rate> ::= {AUTO <sample_rate>} <sample_rate> ::= desired analog sample rate in NR3 format
:ACquire:SRATe[:ANALog]:AUTO {{0 OFF} {1 ON}} (see page 308)	:ACquire:SRATe[:ANALog]:AUTO? (see page 309)	{0 1}
:ACquire:TYPE <type> (see page 311)	:ACquire:TYPE? (see page 311)	<type> ::= {NORMal AVERage 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.

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:AALias?

N (see [page 1708](#))

Query Syntax :ACQuire:AALias?

The :ACQuire:AALias? query returns the current state of the oscilloscope acquisition anti-alias control. This control can be directly disabled or disabled automatically.

Return Format <value><NL>

<value> ::= {1 | 0}

- See Also**
- ["Introduction to :ACQuire Commands"](#) on page 284
 - [":ACQuire:DAALias"](#) on page 289

:ACQuire:COMPLete

C (see [page 1708](#))

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 284
 - [":ACQuire:TYPE"](#) on page 311
 - [":DIGitize"](#) on page 271
 - [":WAVEform:POINts"](#) on page 1566

Example Code

```
' ACQUIRE_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 49](#), "Programming Examples," starting on page 1717

:ACquire:COUNT

C (see [page 1708](#))

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.

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 284
 - [":ACquire:TYPE"](#) on page 311
 - [":DIGitize"](#) on page 271
 - [":WAVEform:COUNT?"](#) on page 1562

:ACQuire:DAALias

N (see [page 1708](#))

Command Syntax :ACQuire:DAALias <mode>

<mode> ::= {DISable | AUTO}

The :ACQuire:DAALias command sets the disable anti-alias mode of the oscilloscope.

When set to DISable, anti-alias is always disabled. This is good for cases where dithered data is not desired.

When set to AUTO, the oscilloscope turns off anti-alias control as needed. Such cases are when the FFT or differentiate math functions are silent. The :DIGitize command always turns off the anti-alias control as well.

Query Syntax :ACQuire:DAALias?

The :ACQuire:DAALias? query returns the oscilloscope's current disable anti-alias mode setting.

Return Format <mode><NL>

<mode> ::= {DIS | AUTO}

- See Also**
- ["Introduction to :ACQuire Commands"](#) on page 284
 - [":ACQuire:AALias?"](#) on page 286

:ACQuire:DIgitizer

N (see [page 1708](#))

Command Syntax :ACQuire:DIgitizer {{0 | OFF} | {1 | ON}}

The :ACQuire:DIgitizer command turns enables or disables Digitizer mode where analog channel sample rate and memory depth are manually specified:

- OFF – the analog channel sample rate and memory depth are automatically determined by the oscilloscope based on the horizontal time/div setting. This is called Automatic mode, and it is the oscilloscope's default.

This setting is equivalent to sending these two commands:

- :ACQuire:SRATe[:ANALog]:AUTO ON
- :ACQuire:POINts[:ANALog]:AUTO ON

In Automatic mode, the oscilloscope's time per division setting determines the sample rate and memory depth so as to fill the waveform display with data while the oscilloscope is running (continuously making acquisitions). For single acquisitions, the time/division setting still determines the sample rate, but the maximum amount of acquisition memory is used.

- ON – the analog channel sample rate and memory depth are set using the :ACQuire:SRATe[:ANALog] and :ACQuire:POINts[:ANALog] commands. This is called Digitizer mode. Digitizer mode primarily aids external software that controls and combines data from multiple instruments.

This setting is equivalent to sending these two commands:

- :ACQuire:SRATe[:ANALog]:AUTO OFF
- :ACQuire:POINts[:ANALog]:AUTO OFF

In Digitizer mode, you choose the acquisition sample rate and memory depth, and those settings are used even though the captured data may extend way beyond the edges of, or take up just a small portion of, the waveform display, depending on the oscilloscope's time/div setting.

Digitizer mode cannot be used along with these other oscilloscope features: XY and Roll time modes, horizontal Zoom display, time references other than Center, segmented memory, serial decode, digital channels, frequency response analysis, mask test, and the power application. In most cases, enabling one of these features when Digitizer mode is enabled will automatically disable Digitizer mode, and then disabling the feature will automatically reenable Digitizer mode.

The :ACQuire:DIgitizer command is an alias for :ACQuire:MANual.

Query Syntax :ACQuire:DIgitizer?

The :ACQuire:DIgitizer? query returns whether Digitizer mode is off or on.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":ACquire:SRATe\[:ANALog\]:AUTO"](#) on page 308
 - [":ACquire:POINts\[:ANALog\]:AUTO"](#) on page 298
 - [":ACquire:MANual"](#) on page 292
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297

:ACquire:MANual

N (see [page 1708](#))

Command Syntax :ACquire:MANual {{0 | OFF} | {1 | ON}}

The :ACquire:MANual command enables or disables Automatic mode where analog channel sample rate and memory depth are automatically determined based on the horizontal time/div setting:

- OFF – the analog channel sample rate and memory depth are automatically determined by the oscilloscope based on the horizontal time/div setting. This is called Automatic mode, and it is the oscilloscope's default.

This setting is equivalent to sending these two commands:

- :ACquire:SRATe[:ANALog]:AUTO ON
- :ACquire:POINts[:ANALog]:AUTO ON

In Automatic mode, the oscilloscope's time per division setting determines the sample rate and memory depth so as to fill the waveform display with data while the oscilloscope is running (continuously making acquisitions). For single acquisitions, the time/division setting still determines the sample rate, but the maximum amount of acquisition memory is used.

- ON – the analog channel sample rate and memory depth are set using the :ACquire:SRATe[:ANALog] and :ACquire:POINts[:ANALog] commands. This is called Digitizer mode. Digitizer mode primarily aids external software that controls and combines data from multiple instruments.

This setting is equivalent to sending these two commands:

- :ACquire:SRATe[:ANALog]:AUTO OFF
- :ACquire:POINts[:ANALog]:AUTO OFF

In Digitizer mode, you choose the acquisition sample rate and memory depth, and those settings are used even though the captured data may extend way beyond the edges of, or take up just a small portion of, the waveform display, depending on the oscilloscope's time/div setting.

Digitizer mode cannot be used along with these other oscilloscope features: XY and Roll time modes, horizontal Zoom display, time references other than Center, segmented memory, serial decode, digital channels, frequency response analysis, mask test, and the power application. In most cases, enabling one of these features when Digitizer mode is enabled will automatically disable Digitizer mode, and then disabling the feature will automatically reenable Digitizer mode.

The :ACquire:MANual command is an alias for :ACquire:DIGitizer.

Query Syntax :ACquire:MANual?

The :ACquire:MANual? query returns the automatic analog sample rate setting.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":ACquire:SRATe\[:ANALog\]:AUTO"](#) on page 308
 - [":ACquire:POINts\[:ANALog\]:AUTO"](#) on page 298
 - [":ACquire:DIGitizer"](#) on page 290
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297

:ACquire:MANual:POINts?

N (see [page 1708](#))

Query Syntax :ACquire:MANual:POINts?

The :ACquire:MANual:POINts? query returns the **User** memory depth graphical user interface setting (whether Auto memory depth is enabled or not).

To disable (or enable) automatic memory depth determination, use the :ACquire:POINts[:ANALog]:AUTO command.

The actual analog channel memory depth is set and queried using the :ACquire:POINts[:ANALog] command/query.

Return Format <gui_setting><NL>
 <gui_setting> ::= User memory depth setting in NR1 format

- See Also**
- ["Introduction to :ACquire Commands"](#) on page 284
 - [":ACquire:POINts\[:ANALog\]:AUTO"](#) on page 298
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297
 - [":ACquire:MANual:SRATe?"](#) on page 295

:ACQuire:MANual:SRATe?

N (see [page 1708](#))

Query Syntax :ACQuire:MANual:SRATe?

The :ACQuire:MANual:SRATe? query returns the **User** sample rate graphical user interface setting (whether Auto sample rate is enabled or not).

To disable (or enable) automatic sample rate determination, use the :ACQuire:SRATe[:ANALog]:AUTO command.

The actual analog channel sample rate is set and queried using the :ACQuire:SRATe[:ANALog] command/query.

Return Format <gui_setting><NL>

<gui_setting> ::= User sample rate setting in NR3 format

- See Also**
- ["Introduction to :ACQuire Commands"](#) on page 284
 - [":ACQuire:SRATe\[:ANALog\]:AUTO"](#) on page 308
 - [":ACQuire:SRATe\[:ANALog\]"](#) on page 307
 - [":ACQuire:MANual:POINTs?"](#) on page 294

:ACquire:MODE

C (see [page 1708](#))

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 284
 - [":ACquire:TYPE"](#) on page 311

:ACquire:POINts[:ANALog]

C (see [page 1708](#))

Command Syntax :ACquire:POINts[:ANALog] <points>

<points> ::= {AUTO | <points_value>}

<points_value> ::= desired analog memory depth in integer NR1 format

Sets the desired acquisition memory depth.

- AUTO – specifies that the analog channel memory depth be automatically determined. It is the same as sending the ":ACquire:POINts[:ANALog]:AUTO ON" command.
- <points_value> – sets the analog channel memory depth to the specified value. The :ACquire:POINts[:ANALog]:AUTO setting will be OFF (if it was not already OFF).

The actual memory depth used is returned by the :ACquire:POINts[:ANALog]? query.

For :SINGle acquisitions, the requested memory depth is used. When the oscilloscope is running (:RUN command, continuously making acquisitions) or when the :DIGitize command is used, the requested amount of memory is smaller.

Query Syntax :ACquire:POINts[:ANALog] ?

The :ACquire:POINts[:ANALog]? 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 284
 - [":ACquire:POINts\[:ANALog\]:AUTO"](#) on page 298
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307

:ACquire:POINts[:ANALog]:AUTO

N (see [page 1708](#))

Command Syntax :ACquire:POINts[:ANALog]:AUTO {{0 | OFF} | {1 | ON}}

The :ACquire:POINts[:ANALog]:AUTO command enables or disables automatic determination of the analog channel memory depth:

- ON – the analog channel memory depth is automatically determined as described in the following table.
- OFF – the analog channel memory depth is set using the :ACquire:POINts[:ANALog] command.

There are four combinations of the "auto" versus "manual" settings for sample rate and memory depth:

		:ACquire:SRATe:AUTO	
		ON	OFF (manual)
:ACquire:POINts:AUTO	ON	This is called Automatic mode. It is the oscilloscope's default. While the oscilloscope is running (continuously making acquisitions), the oscilloscope's time per division setting determines the sample rate and memory depth so as to fill the screen² with data. For single acquisitions, the time/division setting still determines the sample rate, but the maximum amount of acquisition memory is used. See ":ACquire:MANual" on page 292.	You choose the desired sample rate¹ and the memory depth is adjusted to fill the screen² with data at the current time/div setting. The time/div limits are adjusted so that the display can be filled with data at the specified sample rate without exceeding the memory depth. For single acquisitions, the maximum amount of acquisition memory is used. See ":ACquire:SRATe[:ANALog]" on page 307.
	OFF (manual)	You choose the desired memory depth and the sample rate is calculated to be as fast as possible while still being able to fill the screen with data at the current time/div setting. The specified number of points is acquired when running or when making single acquisitions. See ":ACquire:POINts[:ANALog]" on page 297.	This is called Digitizer mode. You choose the desired acquisition sample rate¹ and memory depth, and the time/div setting is adjusted, if necessary, to fill the screen with data. The specified number of points is acquired when running or when making single acquisitions. The actual sample rate and memory depth used are displayed in the dialog box. Digitizer mode primarily aids external software that controls and combines data from multiple instruments. See ":ACquire:DIGitizer" on page 290.

¹ Rounded up to the nearest valid sample rate. Sample rates must be a power-of-two factor of the maximum sample rate (for example, 3.2 GSa/s, 1.6 GSa/s, 800 MSa/s, etc.).

² At least two screens of data are acquired at 2 μ s/div and below. One screen of data is acquired above 2 μ s/div.

When Manual Values Are Not Available Only the ON selection is allowed for :ACquire:POINts[:ANALog]:AUTO and :ACquire:SRATe[:ANALog]:AUTO when:

- When digital channels are on.
- When the XY or Roll time modes are selected.
- When segmented memory is on.
- When mask test is on.

Query Syntax :ACquire:POINts[:ANALog]:AUTO?

The :ACquire:POINts[:ANALog]:AUTO? query returns the automatic analog memory depth setting.

Return Format <setting><NL>

`<setting> ::= {0 | 1}`

- See Also**
- [":ACquire:MANual"](#) on page 292
 - [":ACquire:DIGitizer"](#) on page 290
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307

:ACquire:POINts:MAX?

N (see [page 1708](#))

Query Syntax :ACquire:POINts:MAX?

The :ACquire:POINts:MAX? query returns the oscilloscope's acquisition points maximum limit.

This query is intended to be used when manual memory depth selection is enabled (for example, when ":ACquire:POINts[:ANALog]:AUTO OFF" or ":ACquire:POINts:ANALog <points_value>" have been sent). Otherwise, when automatic memory depth is used, the absolute maximum number of points possible is always returned.

Return Format <max_points><NL>

<max_points> ::= an integer in NR1 format

- See Also**
- ["Introduction to :ACquire Commands"](#) on page 284
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307
 - [":ACquire:POINts\[:ANALog\]:AUTO"](#) on page 298
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297
 - [":ACquire:MANual"](#) on page 292
 - [":ACquire:DIGitizer"](#) on page 290
 - [":DIGitize"](#) on page 271
 - [":WAVEform:POINts"](#) on page 1566

:ACQuire:SEGMENTed:ANALyze

 (see [page 1708](#))

Command Syntax :ACQuire:SEGMENTed:ANALyze

This command calculates measurement statistics and/or infinite persistence over all segments that have been acquired. It corresponds to the graphical user interface **Analyze Segments** button which appears in the Segmented dialog box.

In order to use this command, the oscilloscope must be stopped and in segmented acquisition mode, with either measurements, infinite persistence, histograms, or protocol listing on.

- See Also
- [":ACQuire:MODE"](#) on page 296
 - [":ACQuire:SEGMENTed:COUNT"](#) on page 303
 - ["Introduction to :ACQuire Commands"](#) on page 284

:ACQUIRE:SEGMENTED:COUNT

N (see [page 1708](#))

Command Syntax :ACQUIRE:SEGMENTED:COUNT <count>

<count> ::= an integer from 2 to 2000 in NR1 format

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.

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 2000 in NR1 format

- See Also**
- [":ACQUIRE:MODE"](#) on page 296
 - [":DIGITIZE"](#) on page 271
 - [":SINGLE"](#) on page 277
 - [":RUN"](#) on page 275
 - [":WAVEFORM:SEGMENTED:COUNT?"](#) on page 1574
 - [":ACQUIRE:SEGMENTED:ANALYZE"](#) on page 302
 - ["Introduction to :ACQUIRE Commands"](#) on page 284

Example Code • ["Example Code"](#) on page 304

:ACquire:SEGMented:INDEX

N (see [page 1708](#))

Command Syntax :ACquire:SEGMented:INDEX <index>

<index> ::= an integer from 1 to 2000 in NR1 format

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.

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 2000 in NR1 format

- See Also**
- [":ACquire:MODE"](#) on page 296
 - [":ACquire:SEGMented:COUNT"](#) on page 303
 - [":DIGitize"](#) on page 271
 - [":SINGLE"](#) on page 277
 - [":RUN"](#) on page 275
 - [":WAVEform:SEGMented:COUNT?"](#) on page 1574
 - [":WAVEform:SEGMented:TTAG?"](#) on page 1575
 - [":ACquire:SEGMented:ANALyze"](#) on page 302
 - ["Introduction to :ACquire Commands"](#) on page 284

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
```

```

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If

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::0x2A8D::0x4704::Unset::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[:ANALog]

N (see [page 1708](#))

Command Syntax :ACQuire:SRATe[:ANALog] <rate>

<rate> ::= {AUTO | <sample_rate>}

<sample_rate> ::= desired analog sample rate in NR3 format

Sets the desired acquisition sample rate:

- AUTO – specifies that the analog channel sample rate be automatically determined. It is the same as sending the ":ACQuire:SRATe[:ANALog]:AUTO ON" command.
- <sample_rate> – sets the analog channel sample rate to the specified value. The :ACQuire:SRATe[:ANALog]:AUTO setting will be OFF (if it was not already OFF).

The actual sample rate used is returned by the :ACQuire:SRATe[:ANALog]? query.

Query Syntax :ACQuire:SRATe[:ANALog]? [MAXimum]

The :ACQuire:SRATe[:ANALog]? 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 284
 - [":ACQuire:SRATe\[:ANALog\]:AUTO"](#) on page 308
 - [":ACQuire:POINts\[:ANALog\]"](#) on page 297

:ACQuire:SRATe[:ANALog]:AUTO

N (see [page 1708](#))

Command Syntax :ACQuire:SRATe[:ANALog]:AUTO {{0 | OFF} | {1 | ON}}

The :ACQuire:SRATe[:ANALog]:AUTO command enables or disables automatic determination of the analog channel sample rate:

- ON – the analog channel sample rate is automatically determined as described in the following table.
- OFF – the analog channel sample rate is set using the :ACQuire:SRATe[:ANALog] command.

There are four combinations of the "auto" versus "manual" settings for sample rate and memory depth:

		:ACquire:SRATe:AUTO	
		ON	OFF (manual)
:ACquire:POINts:AUTO	ON	This is called Automatic mode. It is the oscilloscope's default. While the oscilloscope is running (continuously making acquisitions), the oscilloscope's time per division setting determines the sample rate and memory depth so as to fill the screen² with data. For single acquisitions, the time/division setting still determines the sample rate, but the maximum amount of acquisition memory is used. See ":ACquire:MANual" on page 292.	You choose the desired sample rate¹ and the memory depth is adjusted to fill the screen² with data at the current time/div setting. The time/div limits are adjusted so that the display can be filled with data at the specified sample rate without exceeding the memory depth. For single acquisitions, the maximum amount of acquisition memory is used. See ":ACquire:SRATe[:ANALog]" on page 307.
	OFF (manual)	You choose the desired memory depth and the sample rate is calculated to be as fast as possible while still being able to fill the screen with data at the current time/div setting. The specified number of points is acquired when running or when making single acquisitions. See ":ACquire:POINts[:ANALog]" on page 297.	This is called Digitizer mode. You choose the desired acquisition sample rate¹ and memory depth, and the time/div setting is adjusted, if necessary, to fill the screen with data. The specified number of points is acquired when running or when making single acquisitions. The actual sample rate and memory depth used are displayed in the dialog box. Digitizer mode primarily aids external software that controls and combines data from multiple instruments. See ":ACquire:DIGitizer" on page 290.
¹ Rounded up to the nearest valid sample rate. Sample rates must be a power-of-two factor of the maximum sample rate (for example, 3.2 GSa/s, 1.6 GSa/s, 800 MSa/s, etc.). ² At least two screens of data are acquired at 2 μs/div and below. One screen of data is acquired above 2 μs/div.			

When Manual Values Are Not Available Only the ON selection is allowed for :ACquire:POINts[:ANALog]:AUTO and :ACquire:SRATe[:ANALog]:AUTO when:

- When digital channels are on.
- When the XY or Roll time modes are selected.
- When segmented memory is on.
- When mask test is on.

Query Syntax :ACquire:SRATe [:ANALog] :AUTO?

The :ACquire:SRATe[:ANALog]:AUTO? query returns the automatic analog sample rate setting.

Return Format <setting><NL>

`<setting> ::= {0 | 1}`

- See Also**
- [":ACquire:MANual"](#) on page 292
 - [":ACquire:DIGitizer"](#) on page 290
 - [":ACquire:SRATe\[:ANALog\]"](#) on page 307
 - [":ACquire:POINts\[:ANALog\]"](#) on page 297

:ACquire:TYPE

C (see [page 1708](#))

Command Syntax :ACquire:TYPE <type>
 <type> ::= {NORMal | AVERage | 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).

- PEAK – sets the oscilloscope in the peak detect mode. In this mode, :ACquire:COUNT has no meaning.

The AVERage type can give you extra bits of vertical resolution. See the *User's Guide* for an explanation. When getting waveform data acquired using the AVERage type, 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 | PEAK}

- See Also**
- ["Introduction to :ACquire Commands"](#) on page 284
 - [":ACquire:COUNT"](#) on page 288
 - [":ACquire:MODE"](#) on page 296
 - [":DIGitize"](#) on page 271
 - [":WAVEform:FORMat"](#) on page 1565
 - [":WAVEform:TYPE?"](#) on page 1582
 - [":WAVEform:PREamble?"](#) on page 1570

Example Code

```
' ACQUIRE_TYPE - Sets the acquisition mode, which can be NORMAL,  
' PEAK, or AVERAGE.  
myScope.WriteString ":ACquire:TYPE NORMal"
```

See complete example programs at: [Chapter 49](#), “Programming Examples,” starting on page 1717

9 :BUS<n> Commands

Control all oscilloscope functions associated with buses made up of digital channels. See ["Introduction to :BUS<n> Commands"](#) on page 314.

Table 91 :BUS<n> Commands Summary

Command	Query	Options and Query Returns
:BUS<n>:BIT<m> {{0 OFF} {1 ON}} (see page 315)	:BUS<n>:BIT<m>? (see page 315)	{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 316)	:BUS<n>:BITS? (see page 316)	<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 318)	n/a	<n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:DISPlay {{0 OFF} {1 ON}} (see page 319)	:BUS<n>:DISPlay? (see page 319)	{0 1} <n> ::= 1 or 2; an integer in NR1 format

Table 91 :BUS<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:BUS<n>:LABel <string> (see page 320)	:BUS<n>:LABel? (see page 320)	<string> ::= quoted ASCII string up to 32 characters <n> ::= 1 or 2; an integer in NR1 format
:BUS<n>:MASK <mask> (see page 321)	:BUS<n>:MASK? (see page 321)	<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.

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 1708](#))

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.

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 314
 - [":BUS<n>:BITS"](#) on page 316
 - [":BUS<n>:CLEar"](#) on page 318
 - [":BUS<n>:DISPlay"](#) on page 319
 - [":BUS<n>:LABel"](#) on page 320
 - [":BUS<n>:MASK"](#) on page 321

Example Code

```
' Include digital channel 1 in bus 1:
myScope.WriteString ":BUS1:BIT1 ON"
```

:BUS<n>:BITS

N (see [page 1708](#))

Command Syntax :BUS<n>:BITS <channel_list>, <display>

<channel_list> ::= (@<m>,<m>:<m>, ...) where commas separate bits and colons define bit ranges.

<m> ::= An integer, 0,...,15, defines a digital channel affected by the command.

<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.

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.

Query Syntax :BUS<n>:BITS?

The :BUS<n>:BITS? query returns the definition for the specified bus.

Return Format <channel_list>, <display><NL>

<channel_list> ::= (@<m>,<m>:<m>, ...) where commas separate bits and colons define bit ranges.

<display> ::= {0 | 1}

- See Also**
- ["Introduction to :BUS<n> Commands"](#) on page 314
 - [":BUS<n>:BIT<m>"](#) on page 315
 - [":BUS<n>:CLear"](#) on page 318
 - [":BUS<n>:DISPlay"](#) on page 319
 - [":BUS<n>:LABel"](#) on page 320
 - [":BUS<n>:MASK"](#) on page 321
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713
 - ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

Example Code

```
' 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"
```



```
' 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 1708](#))

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.

- See Also**
- ["Introduction to :BUS<n> Commands"](#) on page 314
 - [":BUS<n>:BIT<m>"](#) on page 315
 - [":BUS<n>:BITS"](#) on page 316
 - [":BUS<n>:DISPlay"](#) on page 319
 - [":BUS<n>:LABel"](#) on page 320
 - [":BUS<n>:MASK"](#) on page 321

:BUS<n>:DISPlay

N (see [page 1708](#))

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.

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 314
 - [":BUS<n>:BIT<m>"](#) on page 315
 - [":BUS<n>:BITS"](#) on page 316
 - [":BUS<n>:CLEAr"](#) on page 318
 - [":BUS<n>:LABel"](#) on page 320
 - [":BUS<n>:MASK"](#) on page 321

:BUS<n>:LABel

N (see [page 1708](#))

Command Syntax :BUS<n>:LABel <quoted_string>

<quoted_string> ::= any series of 32 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

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.

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 32 or less characters as a quoted ASCII string.

- See Also**
- ["Introduction to :BUS<n> Commands"](#) on page 314
 - [":BUS<n>:BIT<m>"](#) on page 315
 - [":BUS<n>:BITS"](#) on page 316
 - [":BUS<n>:CLEar"](#) on page 318
 - [":BUS<n>:DISPlay"](#) on page 319
 - [":BUS<n>:MASK"](#) on page 321
 - [":CHANnel<n>:LABel"](#) on page 354
 - [":DISPlay:LABList"](#) on page 439
 - [":DIGital<d>:LABel"](#) on page 401

Example Code

```
' Set the bus 1 label to "Data":
myScope.WriteString ":BUS1:LABel 'Data''
```

:BUS<n>:MASK

N (see [page 1708](#))

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.

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 314
 - [":BUS<n>:BIT<m>"](#) on page 315
 - [":BUS<n>:BITS"](#) on page 316
 - [":BUS<n>:CLEar"](#) on page 318
 - [":BUS<n>:DISPlay"](#) on page 319
 - [":BUS<n>:LABel"](#) on page 320
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

10 :CALibrate Commands

Utility commands for viewing calibration status and for starting the user calibration procedure. See "[Introduction to :CALibrate Commands](#)" on page 324.

Table 92 :CALibrate Commands Summary

Command	Query	Options and Query Returns
n/a	:CALibrate:DATE? (see page 325)	<return value> ::= <year>,<month>,<day>; all in NR1 format
:CALibrate:LABel <string> (see page 326)	:CALibrate:LABel? (see page 326)	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:OUTPut <signal> (see page 327)	:CALibrate:OUTPut? (see page 327)	<signal> ::= {TRIGgers MASK WAVEgen WGEN1 TSource} Note: WAVE and WGEN1 are equivalent.
:CALibrate:PROTECTED { {0 OFF} {1 ON} } (see page 328)	:CALibrate:PROTECTED? (see page 328)	{ "PROTECTED" "UNPROTECTED" }
:CALibrate:START (see page 329)	n/a	n/a
n/a	:CALibrate:STATUS? (see page 330)	<return value> ::= <status_code>,<status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:TEMPerature? (see page 331)	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see page 332)	<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 1708](#))

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 324

:CALibrate:LABel

N (see [page 1708](#))

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> ::= quoted ASCII string of up to 32 characters in length

See Also • ["Introduction to :CALibrate Commands"](#) on page 324

:CALibrate:OUTPut

N (see [page 1708](#))

Command Syntax :CALibrate:OUTPut <signal>

<signal> ::= {TRIGgers | MASK | WAVEgen | WGEN1 | TSource}

Note: WAVE and WGEN1 are equivalent.

The CALibrate:OUTPut command sets the signal that is available on the rear panel AUX OUT BNC:

- TRIGgers – pulse when a trigger event occurs. The output level is 0–5 V into an open circuit, and 0–2.5 V into 50 Ω .
- 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

- TSource – The raw trigger signal from the oscilloscope's trigger circuit is output to Aux Out. It produces a rising edge whenever the input source would cause a trigger, even though that might occur multiple times within the time of a single acquisition. The trigger source can be a front panel analog input channel or an external trigger input. The output level is 0–5 V into an open circuit, and 0–2.5 V into 50 Ω . This option is not available in all trigger modes.

Query Syntax :CALibrate:OUTPut?

The :CALibrate:OUTPut query returns the current source of the AUX OUT BNC signal.

Return Format <signal><NL>

<signal> ::= {TRIG | MASK | WAVE | TSO}

- See Also**
- ["Introduction to :CALibrate Commands"](#) on page 324
 - [":WGEN<w>:FUNction"](#) on page 1607

:CALibrate:PROTECTED

N (see [page 1708](#))

Command Syntax `:CALibrate:PROTECTED {{0 | OFF} | {1 | ON}}`

The :CALibrate:PROTECTED command disables or enables the Cal Protect control which allows user calibration to be run or prevents user calibration from being run.

Query Syntax `:CALibrate:PROTECTED?`

The :CALibrate:PROTECTED? query returns the calibration protect (Cal Protect) control state. The value "PROTECTED" indicates user calibration will be prevented, and "UNPROTECTED" indicates calibration will be allowed.

Return Format `<control><NL>`

`<control> ::= { "PROTECTED" | "UNPROTECTED" }`

- See Also**
- ["Introduction to :CALibrate Commands"](#) on page 324
 - [":CALibrate:START"](#) on page 329

:CALibrate:START

N (see [page 1708](#))

Command Syntax :CALibrate:START

The CALibrate:START command starts the user calibration procedure.

NOTE

Before starting the user calibration procedure, you must clear the Cal Protect control (see :CALibrate:PROTECTED). See the *User's Guide* for more details on user calibration.

-
- See Also**
- ["Introduction to :CALibrate Commands"](#) on page 324
 - [":CALibrate:PROTECTED"](#) on page 328

:CALibrate:STATus?

N (see [page 1708](#))

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 324

:CALibrate:TEMPerature?

N (see [page 1708](#))

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 324

:CALibrate:TIME?

N (see [page 1708](#))

Query Syntax :CALibrate:TIME?

The :CALibrate:TIME? query returns the time of the last calibration.

Return Format <time><NL>

<time> ::= hour,minute,second in NR1 format

See Also • ["Introduction to :CALibrate Commands"](#) on page 324

11 :CGRade Commands

The :CGRade commands control the color grade waveform analysis feature.

When color grade analysis is enabled, a database is created that has the same number of memory locations as there are pixel locations in the waveform viewing area. Each database memory location is a counter. These counters are incremented when sampled waveform data crosses (or "hits") the associated pixel.

When the oscilloscope is stopped, the counter database remains intact until the acquired data display is modified, usually horizontally or vertically. At that time, the database is reset and the last displayed acquisition is used to start a new database.

Table 93 :CGRade Commands Summary

Command	Query	Options and Query Returns
:CGRade:ANALog:ALL { {0 OFF} {1 ON} } (see page 335)	:CGRade:ANALog:ALL? (see page 335)	{ 0 1 }
:CGRade:INVert { {0 OFF} {1 ON} } (see page 336)	:CGRade:INVert? (see page 336)	{ 0 1 }
:CGRade:KEY:SOURce <source> (see page 337)	:CGRade:KEY:SOURce? (see page 337)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMoRY<r>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1-4 in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format
n/a	:CGRade:LEVelS? (see page 338)	<levels> ::= comma-delimited list of color range levels
:CGRade:LEVelS:DISPla y { {0 OFF} {1 ON} } (see page 339)	:CGRade:LEVelS:DISPla y? (see page 339)	{ 0 1 }

Table 93 :CGRade Commands Summary (continued)

Command	Query	Options and Query Returns
:CGRade:MATH:SOURce:STATE {{0 OFF} {1 ON}} (see page 340)	:CGRade:MATH:SOURce:STATE? (see page 340)	{0 1}
:CGRade:RESet (see page 341)	n/a	n/a
:CGRade:SCHeMe <theme> (see page 342)	:CGRade:SCHeMe? (see page 342)	<theme> ::= {TEMPerature INTensity VIRidis HOT ABYSS SKY INFerno}
:CGRade:SOURce <source> (see page 343)	:CGRade:SOURce? (see page 343)	<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
:CGRade:STATe {{0 OFF} {1 ON}} (see page 344)	:CGRade:STATe? (see page 344)	{0 1}

:CGRade:ANALog:ALL

N (see [page 1708](#))

Command Syntax :CGRade:ANALog:ALL {{0 | OFF} | {1 | ON}}

The :CGRade:ANALog:ALL command enables or disables color grade analysis being performed on all analog input channel waveforms.

When analysis on all analog input channels is enabled, you can use the :CGRade:MATH:SOURce:STATe command and the :CGRade:SOURce command to select one other math function or reference waveform to include in the color grade analysis.

NOTE

Analog input channel color grade analysis is performed in hardware and is faster than math function or reference waveform color grade analysis which is performed in software.

Query Syntax :CGRade:ANALog:ALL?

The :CGRade:ANALog:ALL? query returns the current setting.

Return Format <off_on><NL>

<off_on> ::= {0 | 1}

- See Also**
- [":CGRade:MATH:SOURce:STATe"](#) on page 340
 - [":CGRade:SOURce"](#) on page 343

:CGRade:INVert

N (see [page 1708](#))

Command Syntax :CGRade:INVert {{0 | OFF} | {1 | ON}}

The :CGRade:INVert command reverses the order of colors in the scheme.

Query Syntax :CGRade:INVert?

The :CGRade:INVert? query returns whether the selected scheme's colors are reversed.

Return Format <off_on><NL>

{0 | 1}

- See Also**
- [":CGRade:SOURce"](#) on page 343
 - [":CGRade:SCHEME"](#) on page 342
 - [":CGRade:LEVelS"](#) on page 338
 - [":CGRade:RESet"](#) on page 341
 - [":CGRade:LEVelS:DISPlay"](#) on page 339

:CGRade:KEY:SOURce

N (see [page 1708](#))

- Command Syntax** :CGRade:KEY:SOURce <source>
- <source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | WMEMory<r>}
- <n> ::= 1 to (# analog channels) in NR1 format
- <m> ::= 1-4 in NR1 format
- <r> ::= 1 to (# ref waveforms) in NR1 format
- When color grade analysis on all analog input channels is enabled (see :CGRade:ANALog:ALL) and the **Show Color Key** control is selected, the :CGRade:KEY:SOURce command specifies which waveform's database counts are displayed in the color key.
- Query Syntax** :CGRade:KEY:SOURce?
- The :CGRade:KEY:SOURce? query returns the currently selected color grade key waveform source.
- Return Format** <source><NL>
- See Also** • [":CGRade:ANALog:ALL"](#) on page 335

:CGRade:LEVELs

N (see [page 1708](#))

Query Syntax :CGRade:LEVELs?

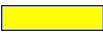
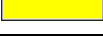
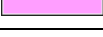
The :CGRade:LEVELs? query returns a comma-delimited list of the hit ranges for each color bin. For example:

```
"1,20,21,40,41,81,82,163,164,326,327,653,654,1307"
```

The first two values show the hit range for the color bin with fewest number of hits, the next two values show the hit range for the next color bin, and so on.

The color used for the greatest number of hits (white in the INTensity theme or TEMPerature theme) represents pixels with the greatest number to about half that number. The next color represents half as many hits, and so on, until the last range (green in the INTensity theme or dark blue in the TEMPerature theme) represents the remaining range of the fewest number of hits.

The following table shows the approximate counter ranges of each color in the INTensity and TEMPerature color grade themes.

:CGRade:SCHEME (for example)		Range (MH = most hits)		
INTensity	TEMPerature	Upper Limit	to	Lower Limit (0 if Upper Limit is 0, Else)
		MH	to	MH/2
		MH/2	to	MH/4
		MH/4	to	MH/8
		MH/8	to	MH/16
		MH/16	to	MH/32
		MH/32	to	MH/64
		MH/64	to	1

Return Format <levels><NL>

<levels> ::= comma-delimited list of hit ranges for each color bin

- See Also**
- [":CGRade:STATE"](#) on page 344
 - [":CGRade:SOURce"](#) on page 343
 - [":CGRade:SCHEME"](#) on page 342
 - [":CGRade:RESet"](#) on page 341
 - [":CGRade:LEVELs:DISPlay"](#) on page 339

:CGRade:LEVels:DISPlay

N (see [page 1708](#))

Command Syntax :CGRade:LEVels:DISPlay {{0 | OFF} | {1 | ON}}

The :CGRade:LEVels:DISPlay command enables or disables the display of the color grade key and levels.

Query Syntax :CGRade:LEVels:DISPlay?

The :CGRade:LEVels:DISPlay? query returns whether the color grade key and levels display is enabled.

Return Format <off_on><NL>
{0 | 1}

- See Also**
- [":CGRade:STATE"](#) on page 344
 - [":CGRade:LEVels"](#) on page 338
 - [":CGRade:SOURce"](#) on page 343
 - [":CGRade:SCHeme"](#) on page 342
 - [":CGRade:RESet"](#) on page 341

:CGRade:MATH:SOURce:STaTe

N (see [page 1708](#))

Command Syntax :CGRade:MATH:SOURce:STaTe {{0 | OFF} | {1 | ON}}

When color grade analysis on all analog input channels is enabled (see :CGRade:ANALog:ALL), the :CGRade:MATH:SOURce:STaTe command enables or disables the addition of one other math function or reference waveform to the color grade analysis. When enabled, use the :CGRade:SOURce command to specify the math function or reference waveform.

NOTE

Analog input channel color grade analysis is performed in hardware and is faster than math function or reference waveform color grade analysis which is performed in software.

Query Syntax :CGRade:MATH:SOURce:STaTe?

The :CGRade:MATH:SOURce:STaTe? query returns the current setting.

Return Format <off_on><NL>

<off_on> ::= {0 | 1}

- See Also**
- [":CGRade:ANALog:ALL"](#) on page 335
 - [":CGRade:SOURce"](#) on page 343

:CGRade:RESet

N (see [page 1708](#))

Command Syntax :CGRade:RESet

The :CGRade:RESet command zeros the color grade counters.

- See Also**
- [":CGRade:STaTe"](#) on page 344
 - [":CGRade:LEVelS"](#) on page 338
 - [":CGRade:SOURce"](#) on page 343
 - [":CGRade:SCHeMe"](#) on page 342
 - [":CGRade:LEVelS:DISPlay"](#) on page 339

:CGRade:SCHEME

N (see [page 1708](#))

Command Syntax :CGRade:SCHEME <theme>

<theme> ::= {TEMPerature | INTensity | VIRidis | HOT | ABYSs | SKY
| INFerno}

The :CGRade:SCHEME command sets the color theme used by color graded waveform display.

The color grade theme specifies seven colors that represent the ranges of counter values (hits) in the database.

Query Syntax :CGRade:SCHEME?

The :CGRade:SCHEME? query returns the currently set color theme.

Return Format <theme><NL>

<theme> ::= {TEMP | INT | VIR | HOT | ABYS | SKY | INF}

- See Also**
- [":CGRade:STATe"](#) on page 344
 - [":CGRade:SOURce"](#) on page 343
 - [":CGRade:LEVelS"](#) on page 338
 - [":CGRade:INVert"](#) on page 336
 - [":CGRade:RESet"](#) on page 341
 - [":CGRade:LEVelS:DISPlay"](#) on page 339

:CGRade:SOURce

N (see [page 1708](#))

Command Syntax :CGRade: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

When color grade analysis on all analog input channels is disabled (see :CGRade:ANALog:ALL), the :CGRade:SOURce command specifies the waveform on which to perform the color grade analysis.

When color grade analysis on all analog input channels is enabled (see :CGRade:ANALog:ALL) and the addition of one other math function or reference waveform is enabled (see :CGRade:MATH:SOURce:STATe), the :CGRade:SOURce command specifies the math function or reference waveform.

Query Syntax :CGRade:SOURce?

The :CGRade:SOURce? query returns the currently specified waveform source.

Return Format <source><NL>

- See Also**
- [":CGRade:STATe"](#) on page 344
 - [":CGRade:SCHeMe"](#) on page 342
 - [":CGRade:LEVels"](#) on page 338
 - [":CGRade:RESet"](#) on page 341
 - [":CGRade:LEVels:DISPlay"](#) on page 339
 - [":CGRade:MATH:SOURce:STATe"](#) on page 340
 - [":CGRade:ANALog:ALL"](#) on page 335

:CGRade:STaTe

N (see [page 1708](#))

Command Syntax :CGRade:STaTe {{0 | OFF} | {1 | ON}}

The :CGRade:STaTe command enables or disables the color grade waveform display.

Query Syntax :CGRade:STaTe?

The :CGRade:STaTe? query returns whether the color grade display is enabled.

Return Format <off_on><NL>

{0 | 1}

- See Also**
- [":CGRade:SOURce"](#) on page 343
 - [":CGRade:SCHeMe"](#) on page 342
 - [":CGRade:LEVels"](#) on page 338
 - [":CGRade:RESet"](#) on page 341
 - [":CGRade:LEVels:DISPlay"](#) on page 339

12 :CHANnel<n> Commands

Control all oscilloscope functions associated with individual analog channels or groups of channels. See **"Introduction to :CHANnel<n> Commands"** on page 348.

Table 94 :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BANDwidth <channel_bw_limit> (see page 349)	:CHANnel<n>:BANDwidth? (see page 349)	{1000000000 350000000 200000000 100000000 50000000 20000000 10000000 5000000} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:COUpling <coupling> (see page 350)	:CHANnel<n>:COUpling? (see page 350)	<coupling> ::= {AC DC} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:DISPlay { {0 OFF} {1 ON} } (see page 351)	:CHANnel<n>:DISPlay? (see page 351)	{0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:IMPedance <impedance> (see page 352)	:CHANnel<n>:IMPedance? (see page 352)	<impedance> ::= {ONEMeg FIFTy} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:INVert { {0 OFF} {1 ON} } (see page 353)	:CHANnel<n>:INVert? (see page 353)	{0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:LABel <string> (see page 354)	:CHANnel<n>:LABel? (see page 354)	<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 page 355)	:CHANnel<n>:OFFSet? (see page 355)	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V mV} <n> ::= 1-2 or 1-4; in NR1 format

Table 94 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe <attenuation> (see page 356)	:CHANnel<n>:PROBe? (see page 356)	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe:BTN <setting> (see page 357)	:CHANnel<n>:PROBe:BTN ? (see page 357)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {HEADlight INFinimode RSTop SINGLE CDISplay AUToscale FTRigger QACTion NACTion NONE}
:CHANnel<n>:PROBe:CAL ibration (see page 358)	n/a	n/a
n/a	:CHANnel<n>:PROBe:CAL ibration:VALidity? (see page 359)	<n> ::= 1 to (# analog channels) in NR1 format <validity> ::= {Cal Valid Cal Invalid}
:CHANnel<n>:PROBe:EXT ernal {{0 OFF} {1 ON}} (see page 360)	:CHANnel<n>:PROBe:EXT ernal? (see page 360)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {0 1}
:CHANnel<n>:PROBe:EXT ernal:GAIN <gain_factor> (see page 361)	:CHANnel<n>:PROBe:EXT ernal:GAIN? (see page 361)	<n> ::= 1 to (# analog channels) in NR1 format <gain_factor> ::= a real number from 0.0001 to 1000 in NR3 format
:CHANnel<n>:PROBe:EXT ernal:OFFSet <offset_value> (see page 362)	:CHANnel<n>:PROBe:EXT ernal:OFFSet? (see page 362)	<n> ::= 1 to (# analog channels) in NR1 format <offset_value> ::= a real number for the offset value at center screen in NR3 format
:CHANnel<n>:PROBe:EXT ernal:UNITs <units> (see page 364)	:CHANnel<n>:PROBe:EXT ernal:UNITs? (see page 364)	<n> ::= 1 to (# analog channels) in NR1 format <units> ::= {VOLT AMPere}
:CHANnel<n>:PROBe:HEA D[:TYPE] <head_param> (see page 365)	:CHANnel<n>:PROBe:HEA D[:TYPE]? (see page 365)	<head_param> ::= {SEND0 SEND6 SEND12 SEND20 DIFF0 DIFF6 DIFF12 DIFF20 DSMA DSMA6 NONE} <n> ::= 1 to (# analog channels) in NR1 format

Table 94 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:CHANnel<n>:PROBe:ID? (see page 366)	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:MODE <setting> (see page 367)	:CHANnel<n>:PROBe:MODE? (see page 367)	<n> ::= 1 to (# analog channels) in NR1 format <setting> ::= {DIFFerential DOFFset SEA SEB CM}
:CHANnel<n>:PROBe:RSE Nse <value> (see page 368)	:CHANnel<n>:PROBe:RSE Nse? (see page 368)	<value> ::= Ohms in NR3 format <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:SKE W <skew_value> (see page 369)	:CHANnel<n>:PROBe:SKE W? (see page 369)	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1 to (# analog channels) in NR1 format
n/a	:CHANnel<n>:PROBe:STATUS? (see page 370)	<n> ::= 1 to (# analog channels) in NR1 format <status> ::= {<PS000XA_status> <N2893A_status>} <PS000XA_status> ::= {Laser Error Sensor Head Error Calibrating OK} <N2893A_status> ::= {Calibrating OK}
:CHANnel<n>:PROBe:STYPe <signal type> (see page 371)	:CHANnel<n>:PROBe:STYPe? (see page 371)	<signal type> ::= {DIFFerential SINGLE ISOLated} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROBe:ZOOM {{0 OFF} {1 ON}} (see page 372)	:CHANnel<n>:PROBe:ZOOM? (see page 372)	<setting> ::= {0 1} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:PROTection (see page 373)	:CHANnel<n>:PROTection? (see page 373)	{NORM TRIP} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:RANGe <range>[suffix] (see page 374)	:CHANnel<n>:RANGe? (see page 374)	<range> ::= Vertical full-scale range value in NR3 format [suffix] ::= {V mV} <n> ::= 1 to (# analog channels) in NR1 format

Table 94 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:SCALe <scale>[suffix] (see page 375)	:CHANnel<n>:SCALe? (see page 375)	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V mV} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:UNITs <units> (see page 376)	:CHANnel<n>:UNITs? (see page 376)	<units> ::= {VOLT AMPere} <n> ::= 1 to (# analog channels) in NR1 format
:CHANnel<n>:VERNier { {0 OFF} {1 ON} } (see page 377)	:CHANnel<n>:VERNier? (see page 377)	{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 32 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 +0.0E+00;COUP DC;IMP ONEM;DISP 1;
BAND +5.00000000E+08;INV 0;LAB "1";UNIT VOLT;PROB +10.000000E+00;
PROB:SKEW +0.0E+00;STYP SING;EXT 0;EXT:GAIN +1.0000E+00
```


:CHANnel<n>:BANDwidth

C (see [page 1708](#))**Command Syntax** :CHANnel<n>:BANDwidth <channel_bw_limit>

<channel_bw_limit> ::= value in NR3 format

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:BANDwidth command lets you limit the bandwidth of the channel to remove unwanted high frequency noise from the waveform and potentially increase the number of bits of vertical resolution. You can select these bandwidths:

- 500000000 or 1000000000 – Maximum (1 GHz) / 14 Bits
- 350000000 – 350 MHz / 14 Bits
- 200000000 – 200 MHz / 14 Bits
- 100000000 – 100 MHz / 15 Bits HD
- 50000000 – 50 MHz / 16 Bits HD
- 20000000 – 20 MHz / 16 Bits HD
- 10000000 – 10 MHz / 16 Bits HD, applied to all analog input channels
- 5000000 – 5 MHz / 16 Bits HD, applied to all analog input channels

The selections marked with "HD" (high-definition) improve the number of bits of vertical resolution.

NOTE

The 5 MHz and 10 MHz selections are automatically applied to all analog input channels.

Query Syntax :CHANnel<n>:BANDwidth?

The :CHANnel<n>:BANDwidth? query returns the current setting of the channel bandwidth limit.

Return Format <channel_bw_limit><NL>

<channel_bw_limit> ::= {1000000000 | 350000000 | 200000000 | 100000000
| 50000000 | 20000000 | 10000000 | 5000000} in NR3 format

See Also • ["Introduction to :CHANnel<n> Commands"](#) on page 348

:CHANnel<n>:COUpling

C (see [page 1708](#))

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 348

:CHANnel<n>:DISPlay

C (see [page 1708](#))

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 348
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270
 - [":STATus?"](#) on page 278
 - [":POD<n>:DISPlay"](#) on page 831
 - [":DIGital<d>:DISPlay"](#) on page 400

:CHANnel<n>:IMPedance

C (see [page 1708](#))

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 M Ω) and FIFTy (50 Ω).

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 348

:CHANnel<n>:INVert

N (see [page 1708](#))

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 348

:CHANnel<n>:LABel

N (see [page 1708](#))

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.

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 348
 - [":DISPlay:LABel"](#) on page 438
 - [":DIGital<d>:LABel"](#) on page 401
 - [":DISPlay:LABList"](#) on page 439
 - [":BUS<n>:LABel"](#) on page 320

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 49](#), "Programming Examples," starting on page 1717

:CHANnel<n>:OFFSet

C (see [page 1708](#))

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 348
 - [":CHANnel<n>:RANGe"](#) on page 374
 - [":CHANnel<n>:SCALE"](#) on page 375
 - [":CHANnel<n>:PROBe"](#) on page 356

:CHANnel<n>:PROBe

C (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe <attenuation>

<attenuation> ::= probe attenuation ratio in NR3 format

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe command specifies the probe attenuation factor for the selected channel.

The probe attenuation factor may be from 0.001 to 10000.

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.

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 :CHANnel<n>:PROBe?

The :CHANnel<n>:PROBe? query returns the current probe attenuation factor for the selected channel.

Return Format <attenuation><NL>

<attenuation> ::= probe attenuation ratio in NR3 format

- See Also**
- ["Introduction to :CHANnel<n> Commands"](#) on page 348
 - [":CHANnel<n>:RANGe"](#) on page 374
 - [":CHANnel<n>:SCALE"](#) on page 375
 - [":CHANnel<n>:OFFSet"](#) on page 355

Example Code

```
' CHANNEL_PROBE - Sets the probe attenuation factor for the
' selected channel. The probe attenuation factor may be set
' from 0.001 to 10000.
myScope.WriteString ":CHANnel1:PROBe 10" ' Set Probe to 10:1.
```

See complete example programs at: [Chapter 49](#), "Programming Examples," starting on page 1717

:CHANnel<n>:PROBe:BTN

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:BTN <setting>

<setting> ::= {HEADlight | INFiniiMode | RSTop | SINGLE | CDISplay
| AUToscale | FTRigger | QACTion | NACTion | NONE}

<n> ::= 1 to (# analog channels) in NR1 format

For N275xA InfiniiMode probes, the :CHANnel<n>:PROBe:BTN command sets the probe's quick-action button mode:

- HEADlight – Pressing the probe's quick-action button toggles the probe headlight on or off. Holding the quick-action button adjusts the intensity of the probe's headlight.
- INFiniiMode – Cycles through the InfiniiMode settings (see :CHANnel<n>:PROBe:MODE).
- RSTop (Run/Stop) – Toggles between continuous data acquisition (running) and stopping data acquisitions (same as the oscilloscope's **[Run/Stop]** key).
- SINGLE – Performs a single acquisition on the oscilloscope (same as the oscilloscope's **[Single]** key).
- CDISplay – Clears the oscilloscope display.
- AUToscale – Performs an oscilloscope Autoscale operation.
- FTRigger (Force Trigger) – Performs an oscilloscope "force trigger".
- QACTion (Quick Action) – Performs the quick action configured on the oscilloscope (see the oscilloscope's *user's guide* for more information).
- NACTion (No Action) – Disables the probe's quick-action button.
- NONE – When a probe does not have a quick-action button, this is the only valid setting.

Query Syntax :CHANnel<n>:PROBe:BTN?

The :CHANnel<n>:PROBe:BTN? query returns the N275xA InfiniiMode probe's quick-action button setting.

Return Format <setting><NL>

<setting> ::= {HEAD | INF | RST | SING | CDIS | AUT | FTR | QACT
| NACT | NONE}

See Also • [":CHANnel<n>:PROBe:MODE"](#) on page 367

:CHANnel<n>:PROBe:CALibration

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:CALibration

For the N7026A and N2893A probes, the :CHANnel<n>:PROBe:CALibration command begins the probe degauss operation.

For the PS000XA probes, the :CHANnel<n>:PROBe:CALibration command performs probe calibration.

- See Also**
- [":CHANnel<n>:PROBe:CALibration:VALidity"](#) on page 359
 - [":CHANnel<n>:PROBe:STATus?"](#) on page 370

:CHANnel<n>:PROBe:CALibration:VALidity

N (see [page 1708](#))

Query Syntax :CHANnel<n>:PROBe:CALibration:VALidity?

<n> ::= 1 to (# analog channels) in NR1 format

For the PS000XA probes only, the :CHANnel<n>:PROBe:CALibration:VALidity? query returns whether the calibration factors for the probe are still valid.

Return Format <validity><NL>

<validity> ::= {Cal Valid | Cal Invalid}

- See Also**
- [":CHANnel<n>:PROBe:CALibration"](#) on page 358
 - [":CHANnel<n>:PROBe:STATus?"](#) on page 370

:CHANnel<n>:PROBe:EXTeRnal

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:EXTeRnal {{0 | OFF} | {1 | ON}}

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:EXTeRnal command enables or disables External Scaling on the analog input channel. External Scaling can apply additional gain to the input channel to account for additional attenuators, adapters, etc., in the probing system.

Query Syntax :CHANnel<n>:PROBe:EXTeRnal?

The :CHANnel<n>:PROBe:EXTeRnal? query returns whether External Scaling is enabled (1) or disabled (0).

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":CHANnel<n>:PROBe:EXTeRnal:GAIN"](#) on page 361
 - [":CHANnel<n>:PROBe:EXTeRnal:UNITs"](#) on page 364

:CHANnel<n>:PROBe:EXTeRnal:GAIN

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:EXTeRnal:GAIN <gain_factor>

<gain_factor> ::= a real number from 0.0001 to 1000 in NR3 format

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:EXTeRnal:GAIN command sets the gain associated with External Scaling.

Query Syntax :CHANnel<n>:PROBe:EXTeRnal:GAIN?

The :CHANnel<n>:PROBe:EXTeRnal:GAIN? query returns the External Scaling gain setting.

Return Format <gain_factor><NL>

- See Also**
- [":CHANnel<n>:PROBe:EXTeRnal"](#) on page 360
 - [":CHANnel<n>:PROBe:EXTeRnal:UNITs"](#) on page 364

:CHANnel<n>:PROBe:EXTeRnal:OFFSet

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:EXTeRnal:OFFSet <offset_value>

<offset_value> ::= a real number for the offset value at center screen in NR3 format

<n> ::= 1 to (# analog channels) in NR1 format

NOTE

CHANnel<n>:PROBe:EXTeRnal command must be set to ON before issuing this command or query or this command will have no effect.

The :CHANnel<n>:PROBe:EXTeRnal:OFFSet command sets the external vertical value for the probe that is represented at the center of the display for the selected channel. Offset parameters are probe and vertical scale dependent.

The <offset_value> is usually expressed in volts, but can be in other measurement units, such as amperes, if you have specified other units using the :CHANnel<n>:PROBe:EXTeRnal:UNITs command.

When using the 113xA series probes, the CHANnel<n>:PROBe:STYPe command determines how the offset is applied. When CHANnel<n>:PROBe:STYPe SINGLE is selected, the :CHANnel<n>:PROBe:EXTeRnal:OFFSet command changes the offset value of the probe amplifier. When CHANnel<n>:PROBe:STYPe DIFFerential is selected, the :CHANnel<n>:PROBe:EXTeRnal:OFFSet command changes the offset value of the channel amplifier.

Query Syntax :CHANnel<n>:PROBe:EXTeRnal:OFFSet?

The :CHANnel<n>:PROBe:EXTeRnal:OFFSet? query returns the current external offset value for the probe on the specified channel.

Return Format <offset_value><NL>

Example Code This example sets the external offset for the probe on channel 1 to 0.125 in the current measurement units. Then, the external offset value of the probe on the specified channel is placed in the variable, varOffset, then prints the contents of the variable to the computer's screen.

```
myScope.WriteString ":SYSTem:HEADer OFF"
myScope.WriteString ":CHANnel1:PROBe:EXTeRnal ON"
myScope.WriteString ":CHANnel1:PROBe:EXTeRnal:OFFSet 125E-3"
myScope.WriteString ":CHANnel1:PROBe:EXTeRnal:OFFSet?"
varOffset = myScope.ReadNumber
Debug.Print FormatNumber(varOffset, 0)
```

See Also

- [":CHANnel<n>:PROBe:EXTeRnal"](#) on page 360
- [":CHANnel<n>:PROBe:EXTeRnal:UNITs"](#) on page 364

- **":CHANnel<n>:PROBe:STYPe"** on page 371

:CHANnel<n>:PROBe:EXTeRnal:UNITs

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:EXTeRnal:UNITs <units>

<units> ::= {VOLT | AMPere}

<n> ::= 1 to (# analog channels) in NR1 format

The :CHANnel<n>:PROBe:EXTeRnal: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>:PROBe:EXTeRnal:UNITs?

The :CHANnel<n>:PROBe:EXTeRnal:UNITs? query returns the current units setting for the specified channel.

Return Format <units><NL>

<units> ::= {VOLT | AMP}

- See Also**
- [":CHANnel<n>:PROBe:EXTeRnal"](#) on page 360
 - [":CHANnel<n>:PROBe:EXTeRnal:GAIN"](#) on page 361

:CHANnel<n>:PROBe:HEAD[:TYPE]

C (see [page 1708](#))

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 | DSMA | DSMA6 | 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, 0 dB.
- SEND6 – Single-ended, 6 dB.
- SEND12 – Single-ended, 12 dB.
- SEND20 – Single-ended, 20 dB.
- DIFF0 – Differential, 0 dB.
- DIFF6 – Differential, 6 dB.
- DIFF12 – Differential, 12 dB.
- DIFF20 – Differential, 20 dB.
- DSMA – Differential SMA probe head, 0 dB.
- DSMA6 – Differential SMA probe head, 6 dB.

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 | DSMA | DSMA6 | NONE}
```

- See Also**
- ["Introduction to :CHANnel<n> Commands"](#) on page 348
 - [":CHANnel<n>:PROBe"](#) on page 356
 - [":CHANnel<n>:PROBe:ID?"](#) on page 366
 - [":CHANnel<n>:PROBe:SKEW"](#) on page 369
 - [":CHANnel<n>:PROBe:SType"](#) on page 371

:CHANnel<n>:PROBe:ID?

C (see [page 1708](#))

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 348
- ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

:CHANnel<n>:PROBe:MODE

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:MODE <setting>

<setting> ::= {DIFFerential | DOFFset | SEA | SEB | CM}

<n> ::= 1 to (# analog channels) in NR1 format

For N275xA InfiniiMode probes, the :CHANnel<n>:PROBe:MODE command sets the probe's mode:

- DIFFerential – Differential.
- DOFFset – Differential with Offset.
- SEA – Single-Ended, side A.
- SEB – Single-Ended, side B.
- CM – Common Mode.

Query Syntax :CHANnel<n>:PROBe:MODE?

The :CHANnel<n>:PROBe:MODE? query returns the N275xA InfiniiMode probe's mode setting.

Return Format <setting><NL>

<setting> ::= {DIFF | DOFF | SEA | SEB | CM}

See Also • [":CHANnel<n>:PROBe:BTN"](#) on page 357

:CHANnel<n>:PROBe:RSENse

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:RSENse <value>

<value> ::= Ohms in NR3 format

<n> ::= 1 to (# analog channels) in NR1 format

When the N2820A high-sensitivity current probe is used with the N2825A user-defined R-sense head, the :CHANnel<n>:PROBe:RSENse command specifies the value of the R-sense resistor that is being probed in the device under test (DUT). Supported R-sense resistor values are from 1 m Ω to 1 M Ω .

Query Syntax :CHANnel<n>:PROBe:RSENse?

The :CHANnel<n>:PROBe:RSENse? query returns the R-sense resistor value setting.

When the N2820A high-sensitivity current probe is not attached, the query returns "INF".

When the N2820A high-sensitivity current probe is attached and a special head is attached to the probe, the query returns the value of the special head. For example, if the head is a "20 m Ω " head, the query returns 0.02.

Return Format {<value> | INF}<NL>

<value> ::= Ohms in NR3 format

See Also • [":CHANnel<n>:PROBe:ZOOM"](#) on page 372

:CHANnel<n>:PROBe:SKEW

C (see [page 1708](#))

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 348

:CHANnel<n>:PROBe:STATus?

N (see [page 1708](#))

Query Syntax :CHANnel<n>:PROBe:STATus?

<n> ::= 1 to (# analog channels) in NR1 format

For the PS000XA and N2893A probes only, the :CHANnel<n>:PROBe:STATus? query returns the probe status.

Return Format <status><NL>

<status> ::= {<PS000XA_status> | <N2893A_status>}

<PS000XA_status> ::= {Calibrating | Calibrated | OK | Laser Error | Sensor Head Error}

<N2893A_status> ::= {Calibrating | Calibrated | OK}

- See Also**
- [":CHANnel<n>:PROBe:CALibration"](#) on page 358
 - [":CHANnel<n>:PROBe:CALibration:VALidity"](#) on page 359

:CHANnel<n>:PROBe:STYPe

C (see [page 1708](#))

Command Syntax

NOTE

This command is valid for the 113xA Series and PS000XA probes only.

```
:CHANnel<n>:PROBe:STYPe <signal_type>
<signal_type> ::= {DIFFerential | SINGle | ISOLated}
<n> ::= 1 to (# analog channels) in NR1 format
```

For the 113xA Series probes:

- 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.

For the PS000XA probes:

- ISOLated is the only valid value.

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 | ISOL}
```

- See Also**
- ["Introduction to :CHANnel<n> Commands"](#) on page 348
 - [":CHANnel<n>:OFFSet"](#) on page 355

:CHANnel<n>:PROBe:ZOOM

N (see [page 1708](#))

Command Syntax :CHANnel<n>:PROBe:ZOOM {{0 | OFF} | {1 | ON}}

<n> ::= 1 to (# analog channels) in NR1 format

When the N2820A high-sensitivity current probe is used with both the Primary and Secondary cables, the :CHANnel<n>:PROBe:ZOOM command specifies whether this cable will have the Zoom In waveform (ON) or the Zoom Out waveform (OFF). The other cable will have the opposite waveform.

Query Syntax :CHANnel<n>:PROBe:ZOOM?

The :CHANnel<n>:PROBe:ZOOM? query returns the zoom setting.

Return Format <setting><NL>

<setting> ::= {0 | 1}

See Also • [":CHANnel<n>:PROBe:RSENse"](#) on page 368

:CHANnel<n>:PROTection

N (see [page 1708](#))

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 Ω , 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 Ω .

The :CHANnel<n>:PROTection[:CLEar] command is used to clear (reset) the overload protection. It allows the channel to be used again in 50 Ω mode after the signal that caused the overload has been removed from the channel input.

Reset the analog channel input impedance to 50 Ω (see [":CHANnel<n>:IMPedance"](#) on page 352) 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 348
 - [":CHANnel<n>:COUpling"](#) on page 350
 - [":CHANnel<n>:IMPedance"](#) on page 352
 - [":CHANnel<n>:PROBe"](#) on page 356

:CHANnel<n>:RANGe

C (see [page 1708](#))

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 348
 - [":CHANnel<n>:SCALE"](#) on page 375
 - [":CHANnel<n>:PROBe"](#) on page 356

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 49](#), "Programming Examples," starting on page 1717

:CHANnel<n>:SCALE

N (see [page 1708](#))

Command Syntax :CHANnel<n>:SCALE <scale>[<suffix>]

<scale> ::= vertical units per division in NR3 format

<suffix> ::= {V | mV}

<n> ::= 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.

Query Syntax :CHANnel<n>:SCALE?

The :CHANnel<n>:SCALE? query returns the current scale setting for the specified channel.

Return Format <scale value><NL>

<scale value> ::= vertical units per division in NR3 format

- See Also**
- ["Introduction to :CHANnel<n> Commands"](#) on page 348
 - [":CHANnel<n>:RANGe"](#) on page 374
 - [":CHANnel<n>:PROBe"](#) on page 356

:CHANnel<n>:UNITs

N (see [page 1708](#))

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 348
 - [":CHANnel<n>:RANGe"](#) on page 374
 - [":CHANnel<n>:PROBe"](#) on page 356
 - [":EXternal:UNITs"](#) on page 463

:CHANnel<n>:VERNier

N (see [page 1708](#))

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 348

13 :CLOCK Commands

The :CLOCK commands specify the clock recovery options.

Table 95 :CLOCK Commands Summary

Command	Query	Options and Query Returns
:CLOCK {{0 OFF} {1 ON}} (see page 381)	:CLOCK? (see page 381)	{0 1}
:CLOCK:METHod <method> (see page 382)	:CLOCK:METHod? (see page 383)	<method> ::= {<fixed> <fopll> <sopll> <explicit>} <fixed> ::= FIXed, {AUTO SEMI[, <data_rate>] <data_rate>} <fopll> ::= FOPLL, <data_rate>, <loop_bandwidth> <sopll> ::= SOPLL, <data_rate>, <loop_bandwidth>, <damping_factor> <explicit> ::= EXPLicit, <source>, {RISing FALLing BOTH}[, <multiplier>] <data_rate> ::= the data signal's expected bits/second rate in NR3 format <loop_bandwidth> ::= the -3 dB bandwidth specification of PLL in NR3 format <damping_factor> ::= of the PLL's observed jitter transfer function in NR3 format <source> ::= explicit clock source, CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <multiplier> ::= explicit clock multiplier from 1 to 1000 in NR1 format

:CLOCK

N (see [page 1708](#))

Command Syntax :CLOCK {{0 | OFF} | {1 | ON}}

The :CLOCK command turns the display of the recovered clock on or off.

Math function 1 is used to display the recovered clock.

Query Syntax :CLOCK?

The :CLOCK? query returns whether the recovered clock is currently displayed.

Return Format <off_on><NL>

{0 | 1}

See Also • [":CLOCK:METHod"](#) on page 382

:CLOCK:METHod

N (see [page 1708](#))

Command Syntax :CLOCK:METHod <method>

<method> ::= {<fixed> | <fopll> | <sopll> | <explicit>}

<fixed> ::= FIXed, {AUTO | SEMI[, <data_rate>] | <data_rate>}

<fopll> ::= FOPLl, <data_rate>, <loop_bandwidth>

<sopll> ::= SOPLl, <data_rate>, <loop_bandwidth>, <damping_factor>

<explicit> ::= EXPLicit, <source>, {RISing | FALLing | BOTH}[, <multiplier>]

<data_rate> ::= the data signal's expected bits/second rate in NR3 format

<loop_bandwidth> ::= the -3 dB bandwidth specification of PLL in NR3 format

<damping_factor> ::= of the PLL's observed jitter transfer function in NR3 format

<source> ::= explicit clock source, CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

<multiplier> ::= explicit clock multiplier from 1 to 1000 in NR1 format

The :CLOCK:METHod command sets the clock recovery method:

- **FIXed** – The clock is assumed to be a constant frequency, and the data rate recovery options are:
 - **AUTO** – The clock is recovered using the signal edge database only. The narrowest pulse is presumed to be an isolated one or zero.
 - **SEMI** – The clock is recovered using the expected nominal data rate as a "seed" and the signal edge database.
 - **<data_rate>** – The expected nominal data rate you enter is used. It should be the reciprocal of the signal's unit interval.
- **FOPLl** – With First Order PLL, the clock is recovered using the DSP-modeled response of an ideal first-order hardware PLL.
- **SOPLl** – With Second Order PLL, the clock is recovered using the DSP-modeled response of an ideal Type II, second-order hardware PLL.

Typical damping factor values are 1.0, which is critically damped, and 0.707, which is considered ideal or optimal because of its fast settling time (optimally flat) and Butterworth type response. Values less than 1 are underdamped, and values greater than 1 are overdamped.

- **EXPLicit** – The clock is recovered from a second signal in your device under test, with a multiplier from 1 to 1000 applied.

Query Syntax :CLOCK:METHod?

The :CLOCK:METHod? query returns the currently specified clock recovery method.

Return Format <method><NL>

See Also • **":CLOCK"** on page 381

14 :COUNter<c> Commands

These commands control the counter feature. See ["Introduction to :COUNter<c> Commands"](#) on page 385.

Table 96 :COUNter<c> Commands Summary

Command	Query	Options and Query Returns
n/a	:COUNter<c>:CURRent? (see page 387)	<c> ::= {1 2 3 4} <value> ::= current counter value in NR3 format
:COUNter<c>:ENABle { {0 OFF} {1 ON} } (see page 388)	:COUNter<c>:ENABle? (see page 388)	<c> ::= {1 2 3 4} {0 1}
:COUNter<c>:MODE <mode> (see page 389)	:COUNter<c>:MODE (see page 389)	<c> ::= {1 2 3 4} <mode> ::= {FREQuency PERiod TOTAlize}
:COUNter<c>:NDIGits <value> (see page 390)	:COUNter<c>:NDIGits (see page 390)	<c> ::= {1 2 3 4} <value> ::= 3 to 8 in NR1 format
:COUNter<c>:SOURce <source> (see page 391)	:COUNter<c>:SOURce? (see page 391)	<c> ::= {1 2 3 4} <source> ::= {CHANnel<n> TQEvent} <n> ::= 1 to (# analog channels) in NR1 format
:COUNter<c>:TOTAlize: CLEar (see page 392)	n/a	<c> ::= {1 2 3 4}
:COUNter<c>:TOTAlize: SLOPe <slope> (see page 393)	:COUNter<c>:TOTAlize: SLOPe? (see page 393)	<c> ::= {1 2 3 4} <slope> ::= { {NEGative FALLing} {POSitive RISing} }

Introduction to :COUNter<c> Commands The :COUNter subsystem provides commands to control the counter feature.
Reporting the Setup

Use :COUNter<c>? to query setup information for the COUNter subsystem.

Return Format

The following is a sample response from the :COUNter1? query. In this case, the query was issued following the *RST command.

```
:COUN1:ENAB 0;SOUR CHAN1;MODE FREQ;NDIG 5
```

:COUNter<c>:CURRent?

N (see [page 1708](#))

Query Syntax :COUNter<c>:CURRent?

<c> ::= {1 | 2 | 3 | 4}

The :COUNter<c>:CURRent? query returns the current counter value.

Return Format <value><NL>

<value> ::= current counter value in NR3 format

- See Also**
- [":COUNter<c>:ENABLE"](#) on page 388
 - [":COUNter<c>:MODE"](#) on page 389
 - [":COUNter<c>:NDIGits"](#) on page 390
 - [":COUNter<c>:SOURce"](#) on page 391

:COUNTER<c>:ENABLE

N (see [page 1708](#))

Command Syntax :COUNTER<c>:ENABLE {{0 | OFF} | {1 | ON}}

<c> ::= {1 | 2 | 3 | 4}

The :COUNTER<c>:ENABLE command enables or disables the counter feature.

Query Syntax :COUNTER<c>:ENABLE?

The :COUNTER<c>:ENABLE? query returns whether the counter is enabled or disabled.

Return Format <off_on><NL>

{0 | 1}

- See Also**
- [":COUNTER<c>:CURRENT?"](#) on page 387
 - [":COUNTER<c>:MODE"](#) on page 389
 - [":COUNTER<c>:NDIGits"](#) on page 390
 - [":COUNTER<c>:SOURCE"](#) on page 391

:COUNter<c>:MODE

N (see [page 1708](#))

Command Syntax :COUNter<c>:MODE <mode>

<c> ::= {1 | 2 | 3 | 4}

<mode> ::= {FREQuency | PERiod | TOTalize}

The :COUNter<c>: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<c>:MODE?

The :COUNter<c>:MODE? query returns the counter mode setting.

Return Format <mode><NL>

<mode> ::= {FREQ | PER | TOT}

- See Also**
- [":COUNter<c>:CURRent?"](#) on page 387
 - [":COUNter<c>:ENABle"](#) on page 388
 - [":COUNter<c>:NDIGits"](#) on page 390
 - [":COUNter<c>:SOURce"](#) on page 391
 - [":COUNter<c>:TOTalize:CLEar"](#) on page 392
 - [":COUNter<c>:TOTalize:SLOPe"](#) on page 393

:COUNTER<c>:NDIGits

N (see [page 1708](#))

Command Syntax :COUNTER<c>:NDIGits <value>

<c> ::= {1 | 2 | 3 | 4}

<value> ::= 3 to 8 in NR1 format

The :COUNTER<c>: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<c>:NDIGits?

The :COUNTER<c>: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<c>:CURRENT?"](#) on page 387
 - [":COUNTER<c>:ENABLE"](#) on page 388
 - [":COUNTER<c>:MODE"](#) on page 389
 - [":COUNTER<c>:SOURCE"](#) on page 391
 - [":COUNTER<c>:TOTALize:CLEAR"](#) on page 392
 - [":COUNTER<c>:TOTALize:SLOPe"](#) on page 393

:COUNter<c>:SOURce

N (see [page 1708](#))

Command Syntax :COUNter<c>:SOURce <source>

<c> ::= {1 | 2 | 3 | 4}

<source> ::= {CHANnel<n> | TQEvent}

<n> ::= 1 to (# analog channels) in NR1 format

The :COUNter<c>: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 TQEvent (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 AUX 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<c>:SOURce?

The :COUNter<c>:SOURce? query returns the currently set counter source channel.

Return Format <source><NL>

- See Also**
- [":COUNter<c>:CURRent?"](#) on page 387
 - [":COUNter<c>:ENABLE"](#) on page 388
 - [":COUNter<c>:MODE"](#) on page 389
 - [":COUNter<c>:NDIGits"](#) on page 390

:COUNTER<c>:TOTALize:CLEar

N (see [page 1708](#))

Command Syntax :COUNTER<c>:TOTALize:CLEar

<c> ::= {1 | 2 | 3 | 4}

The :COUNTER<c>:TOTALize:CLEar command zeros the edge event counter.

- See Also**
- [":COUNTER<c>:CURRENT?"](#) on page 387
 - [":COUNTER<c>:ENABLE"](#) on page 388
 - [":COUNTER<c>:MODE"](#) on page 389
 - [":COUNTER<c>:NDIGits"](#) on page 390
 - [":COUNTER<c>:SOURCE"](#) on page 391
 - [":COUNTER<c>:TOTALize:SLOPe"](#) on page 393

:COUNter<c>:TOTalize:SLOPe

N (see [page 1708](#))

Command Syntax :COUNter<c>:TOTalize:SLOPe <slope>

<c> ::= {1 | 2 | 3 | 4}

<slope> ::= {{NEGative | FALLing} | {POSitive | RISing}}

The :COUNter<c>:TOTalize:SLOPe command specifies whether positive or negative edges are counted.

Query Syntax :COUNter<c>:TOTalize:SLOPe?

The :COUNter<c>:TOTalize:SLOPe? query returns the currently set slope specification.

Return Format <slope><NL>

<slope> ::= {NEG | POS}

- See Also**
- [":COUNter<c>:CURRent?"](#) on page 387
 - [":COUNter<c>:ENABle"](#) on page 388
 - [":COUNter<c>:MODE"](#) on page 389
 - [":COUNter<c>:NDIGits"](#) on page 390
 - [":COUNter<c>:SOURce"](#) on page 391
 - [":COUNter<c>:TOTalize:CLEar"](#) on page 392

15 :DIGital<d> Commands

Control all oscilloscope functions associated with individual digital channels. See ["Introduction to :DIGital<d> Commands"](#) on page 396.

Table 97 :DIGital<d> Commands Summary

Command	Query	Options and Query Returns
:DIGital:ORDER:ASCending (see page 397)	n/a	n/a
:DIGital:ORDER:DESCending (see page 398)	n/a	n/a
:DIGital:ORDER[:SET] <digital_channels_list> (see page 399)	:DIGital:ORDER[:SET]? (see page 399)	<digital_channels_list> ::= comma-separated list, e.g., DIGital<d>,DIGital<d>,... <d> ::= 0 to (# digital channels - 1) in NR1 format
:DIGital<d>:DISPlay {{0 OFF} {1 ON}} (see page 400)	:DIGital<d>:DISPlay? (see page 400)	<d> ::= 0 to (# digital channels - 1) in NR1 format {0 1}
:DIGital<d>:LABel <string> (see page 401)	:DIGital<d>:LABel? (see page 401)	<d> ::= 0 to (# digital channels - 1) in NR1 format <string> ::= any series of 32 or less ASCII characters enclosed in quotation marks
n/a	:DIGital<d>:SETup? (see page 402)	<d> ::= 0 to (# digital channels - 1) in NR1 format Returns setup information string, for example, ":DIG0:DISP 1;THR +1.40E+00"

Table 97 :DIGital<d> Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGital<d>:SIZE <value> (see page 403)	:DIGital<d>:SIZE? (see page 403)	<d> ::= 0 to (# digital channels - 1) in NR1 format <value> ::= {SMALl MEDium LARGe}
:DIGital<d>:THReshold <value>[suffix] (see page 404)	:DIGital<d>:THReshold ? (see page 404)	<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}

Introduction to :DIGital<d> Commands

<d> ::= 0 to (# digital channels - 1) in NR1 format

The DIGital subsystem commands control the viewing, labeling, and positioning of digital channels. They also control threshold settings for groups of digital channels, or *pods*.

Reporting the Setup

Use :DIGital<d>:SETup? to query setup information for the DIGital subsystem.

Return Format

The following is a sample response from the :DIGital0:SETup? query. In this case, the query was issued following a *RST command.

```
:DIG0:DISP 0;THR +1.40E+00;LAB "D0"
```


:DIGital:ORDer:ASCending

N (see [page 1708](#))

Command Syntax :DIGital:ORDer:ASCending

The :DIGital:ORDer:ASCending command displays the digital channels that are turned on (see :DIGital<d>:DISPlay) in ascending order.

- See Also**
- [":DIGital<d>:DISPlay"](#) on page 400
 - [":DIGital:ORDer:DESCending"](#) on page 398
 - [":DIGital:ORDer\[:SET\]"](#) on page 399

:DIGital:ORDer:DESCending

N (see [page 1708](#))

Command Syntax :DIGital:ORDer:DESCending

The :DIGital:ORDer:DESCending command displays the digital channels that are turned on (see :DIGital<d>:DISPlay) in descending order.

- See Also**
- [":DIGital<d>:DISPlay"](#) on page 400
 - [":DIGital:ORDer:ASCending"](#) on page 397
 - [":DIGital:ORDer\[:SET\]"](#) on page 399

:DIGital:ORDer[:SET]

N (see [page 1708](#))

Command Syntax :DIGital:ORDer[:SET] <digital_channels_list>

<digital_channels_list> ::= comma-separated list,
e.g., DIGital<d>,DIGital<d>,...

<d> ::= 0 to (# digital channels - 1) in NR1 format

The :DIGital:ORDer[:SET] command displays digital channels in the order listed.

Digital channels not in the list are turned off and are not displayed.

Query Syntax :DIGital:ORDer[:SET]?

The :DIGital:ORDer[:SET]? query returns the current order of displayed digital channels.

Return Format <digital_channels_list><NL>

<digital_channels_list> ::= comma-separated list

- See Also**
- [":DIGital:ORDer:ASCending"](#) on page 397
 - [":DIGital:ORDer:DESCending"](#) on page 398
 - [":DIGital<d>:DISPlay"](#) on page 400

:DIGital<d>:DISPlay

N (see [page 1708](#))

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.

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 396
 - [":DIGital<d>:SETup?"](#) on page 402
 - [":POD<n>:DISPlay"](#) on page 831
 - [":CHANnel<n>:DISPlay"](#) on page 351
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270
 - [":STATus?"](#) on page 278

:DIGital<d>:LABel

N (see [page 1708](#))

Command Syntax :DIGital<d>:LABel <string>

<d> ::= 0 to (# digital channels - 1) in NR1 format

<string> ::= any series of 32 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

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.

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 32 or less characters as a quoted ASCII string.

- See Also**
- ["Introduction to :DIGital<d> Commands"](#) on page 396
 - [":CHANnel<n>:LABel"](#) on page 354
 - [":DISPlay:LABList"](#) on page 439
 - [":BUS<n>:LABel"](#) on page 320

:DIGital<d>:SETup?

N (see [page 1708](#))

Query Syntax :DIGital<d>:SETup?

The :DIGital<d>:SETup? query returns setup information for the specified digital channel.

Return Format <string><NL>

<string> ::= setup information string

For example:

```
:DIG0:DISP 1;THR +1.40E+00
```

- See Also**
- [":DIGital<d>:DISPlay"](#) on page 400
 - [":DIGital<d>:THReshold"](#) on page 404

:DIGital<d>:SIZE

N (see [page 1708](#))

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.

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 396
 - [":POD<n>:SIZE"](#) on page 834
 - [":DIGital:ORDer\[:SET\]"](#) on page 399

:DIGital<d>:THReshold

N (see [page 1708](#))

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).

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 396
 - [":DIGital<d>:SETup?"](#) on page 402
 - [":POD<n>:THReshold"](#) on page 835
 - [":TRIGger\[:EDGE\]:LEVel"](#) on page 1507

16 :DISPlay Commands

Control how waveforms, graticule, and text are displayed and written on the screen. See "[Introduction to :DISPlay Commands](#)" on page 409.

Table 98 :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n> { {0 OFF} {1 ON} } (see page 410)	:DISPlay:ANNotation<n>? (see page 410)	{0 1} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:BACKground <mode> (see page 411)	:DISPlay:ANNotation<n>:BACKground? (see page 411)	<mode> ::= {OPAQue INVerted TRANSPARENT} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:COLor <color> (see page 412)	:DISPlay:ANNotation<n>:COLor? (see page 412)	<color> ::= {CH1 CH2 CH3 CH4 DIG MATH REF MARKer WHITE RED} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:GRID <number> (see page 413)	:DISPlay:ANNotation<n>:GRID? (see page 413)	<number> ::= an integer from 1 to 20 in NR1 format. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:MODE <mode> (see page 414)	:DISPlay:ANNotation<n>:MODE? (see page 414)	<mode> ::= {GRID SOURce} <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:SOURce <source> (see page 415)	:DISPlay:ANNotation<n>:SOURce? (see page 415)	<source> ::= {CHANnel<n> WMEMory<n> FUNCTion<n> FFT DIGital<n>} <n> ::= an integer in NR1 format.
:DISPlay:ANNotation<n>:TEXT <string> (see page 416)	:DISPlay:ANNotation<n>:TEXT? (see page 416)	<string> ::= quoted ASCII string (up to 254 characters) <n> ::= an integer from 1 to 20 in NR1 format.

Table 98 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:ANNotation<n>:XPOsition <value> (see page 417)	:DISPlay:ANNotation<n>:XPOsition? (see page 417)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:ANNotation<n>:YPOsition <value> (see page 418)	:DISPlay:ANNotation<n>:YPOsition? (see page 418)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height. <n> ::= an integer from 1 to 20 in NR1 format.
:DISPlay:BACKlight { {0 OFF} {1 ON} } (see page 419)	n/a	n/a
:DISPlay:CLear (see page 420)	n/a	n/a
:DISPlay:CLOCK:IGRid { {0 OFF} {1 ON} } (see page 421)	:DISPlay:CLOCK:IGRid? (see page 421)	<setting> ::= {0 1}
:DISPlay:CLOCK:IGRid:XPOsition <value> (see page 422)	:DISPlay:CLOCK:IGRid:XPOsition? (see page 422)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.
:DISPlay:CLOCK:IGRid:YPOsition <value> (see page 423)	:DISPlay:CLOCK:IGRid:YPOsition? (see page 423)	<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.
:DISPlay:CLOCK[:STATe]] { {0 OFF} {1 ON} } (see page 424)	:DISPlay:CLOCK[:STATe]]? (see page 424)	<setting> ::= {0 1}
n/a	:DISPlay:DATA? [<format>] (see page 425)	<format> ::= {BMP PNG} <display data> ::= data in IEEE 488.2 # format
:DISPlay:DIALog:CLOSe (see page 426)	n/a	n/a
:DISPlay:DIALog:TRANs parent { {0 OFF} {1 ON} } (see page 427)	:DISPlay:DIALog:TRANs parent? (see page 427)	<setting> ::= {0 1}
:DISPlay:GRATicule:AL ABels { {0 OFF} {1 ON} } (see page 428)	:DISPlay:GRATicule:AL ABels? (see page 428)	<setting> ::= {0 1}

Table 98 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:GRATicule:ALABels:DUAL {{0 OFF} {1 ON}} (see page 429)	:DISPlay:GRATicule:ALABels:DUAL? (see page 429)	<setting> ::= {0 1}
:DISPlay:GRATicule:ALABels:IGrid {{0 OFF} {1 ON}} (see page 430)	:DISPlay:GRATicule:ALABels:IGrid? (see page 430)	<setting> ::= {0 1}
:DISPlay:GRATicule:COUNT <value> (see page 431)	:DISPlay:GRATicule:COUNT? (see page 431)	<num_grids> ::= an integer from 1 to 4 in NR1 format.
:DISPlay:GRATicule:FScreen {{0 OFF} {1 ON}} (see page 432)	:DISPlay:GRATicule:FScreen? (see page 432)	<setting> ::= {0 1}
:DISPlay:GRATicule:INTensity <value> (see page 433)	:DISPlay:GRATicule:INTensity? (see page 433)	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:GRATicule:LAYout <type> (see page 434)	:DISPlay:GRATicule:LAYout? (see page 434)	<type> ::= {STACKed TILED OVERlay CUSTom}
:DISPlay:GRATicule:SE T <source>, <grid> (see page 435)	n/a	<source> ::= {CHANnel<n> WMemory<n> FUNCTION<n> FFT {PBUS<n> SBUS<n>} GAIN PHASE} <grid> ::= a grid number integer from 1 to 4 in NR1 format.
n/a	:DISPlay:GRATicule:SOURce? <grid> (see page 436)	<grid> ::= a grid number integer from 1 to 4 in NR1 format. Returns: <sources> ::= comma-separated list of the waveforms displayed in the grid. Can contain items like: CHAN<n>, WMEM<n>, FUNC<n>, FFT, SBUS<n>, GAIN, PHASE
:DISPlay:INTensity:WAVEform <value> (see page 437)	:DISPlay:INTensity:WAVEform? (see page 437)	<value> ::= an integer from 0 to 100 in NR1 format.
:DISPlay:LAbel {{0 OFF} {1 ON}} (see page 438)	:DISPlay:LAbel? (see page 438)	{0 1}

Table 98 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:LABList <binary block> (see page 439)	:DISPlay:LABList? (see page 439)	<binary block> ::= an ordered list of up to 75 labels, each 10 characters maximum, separated by newline characters
:DISPlay:MESSAge:CLEAr (see page 440)	n/a	n/a
:DISPlay:PERSPistence <value> (see page 441)	:DISPlay:PERSPistence? (see page 441)	<value> ::= {MINimum INFinite <time>} <time> ::= seconds in in NR3 format from 100E-3 to 60E0
:DISPlay:PERSPistence:CLEAr (see page 442)	n/a	n/a
n/a	:DISPlay:RESuLts:CATalog? (see page 443)	<windows> ::= list of the results windows displayed from left to right. Can contain: MEAS, MARK, HIST, DVM, COUN, LIST, MTES, FRAN
:DISPlay:RESuLts:LAYout <format> (see page 444)	:DISPlay:RESuLts:LAYout? (see page 444)	<format> ::= {TAB LIST}
:DISPlay:RESuLts:LAYout:LIST[:SElect] <results_window> (see page 445)	:DISPlay:RESuLts:LAYout:LIST[:SElect]? (see page 445)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}
:DISPlay:RESuLts:LAYout:TAB:LEFt[:SElect] <results_window> (see page 446)	:DISPlay:RESuLts:LAYout:TAB:LEFt[:SElect]? (see page 446)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}
:DISPlay:RESuLts:LAYout:TAB:RIgHt[:SElect] <results_window> (see page 447)	:DISPlay:RESuLts:LAYout:TAB:RIgHt[:SElect]? (see page 447)	<results_window> ::= {MEAS MARK HIST DVM COUN LIST MTES FRAN}
:DISPlay:RESuLts:SIZE <size> (see page 448)	:DISPlay:RESuLts:SIZE? (see page 448)	<size> ::= {CUSTom,<height> FULL HIDDEN} <height> ::= a pixel height integer from 36 to 1024 in NR1 format.
:DISPlay:TRANSPARENT {OFF ON} (see page 449)	:DISPlay:TRANSPARENT? (see page 449)	{OFF ON}

Table 98 :DISPlay Commands Summary (continued)

Command	Query	Options and Query Returns
:DISPlay:VECTors {{0 OFF} {1 ON}} (see page 450)	:DISPlay:VECTors? (see page 450)	{0 1}
:DISPlay:XYColor <xy_color_scheme> (see page 451)	:DISPlay:XYColor? (see page 451)	<xy_color_scheme> ::= {VIRidis HOT ABYSS SKY INFerno}

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.
- 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;PERS MIN
```

:DISPlay:ANNotation<n>

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n> <setting>

<setting> ::= {{1 | ON} | {0 | OFF}}

<n> ::= an integer from 1 to 20 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 416
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:BACKground

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:BACKground <mode>

<mode> ::= {OPAQue | INVerted | TRANSPARENT}

<n> ::= an integer from 1 to 20 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 410
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:COLor

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:COLor <color>

<color> ::= {CH1 | CH2 | CH3 | CH4 | DIG | MATH | REF | MARKer | WHITE
| RED}

<n> ::= an integer from 1 to 20 in NR1 format.

When an annotation's mode is GRID (see :DISPlay:ANNotation<n>:MODE), 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>:MODE"](#) on page 414
 - [":DISPlay:ANNotation<n>:GRID"](#) on page 413
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:GRID

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:GRID <number>

<number> ::= an integer from 1 to 4 in NR1 format.

<n> ::= an integer from 1 to 20 in NR1 format.

When an annotation's mode is GRID (see :DISPlay:ANNotation<n>:MODE), the :DISPlay:ANNotation<n>:GRID command makes the association to a grid.

When an annotation is associated with a grid, you can specify the annotation color using the :DISPlay:ANNotation<n>:COLor command.

Query Syntax :DISPlay:ANNotation<n>:GRID?

The :DISPlay:ANNotation<n>:GRID? query returns the grid to which the annotation is associated.

Return Format <number><NL>

<number> ::= an integer from 1 to 4 in NR1 format.

- See Also**
- [":DISPlay:ANNotation<n>:MODE"](#) on page 414
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:MODE

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:MODE <mode>

<mode> ::= {GRID | SOURce}

<n> ::= an integer from 1 to 4 in NR1 format.

The :DISPlay:ANNotation<n>:MODE command specifies the annotation's association:

- GRID — The annotation is associated with a grid. Use the :DISPlay:ANNotation<n>:GRID command to select the grid.

The annotation will remain in the grid as the grid layout changes.

When an annotation is associated with a grid, you can specify the annotation color using the :DISPlay:ANNotation<n>:COLor command.

- SOURce — The annotation is associated with a waveform source. Use the :DISPlay:ANNotation<n>:SOURce command to select the waveform source.

The annotation will remain with the waveform source as the waveform is moved between grids.

When an annotation is associated with a waveform source, its color is the same as the waveform source.

Query Syntax :DISPlay:ANNotation<n>:MODE?

The :DISPlay:ANNotation<n>:MODE? query returns the annotation mode.

Return Format <mode><NL>

<mode> ::= {GRID | SOUR}

- See Also**
- [":DISPlay:ANNotation<n>:GRID"](#) on page 413
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:SOURce"](#) on page 415
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:SOURce

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:SOURce <source>
 <source> ::= {CHANnel<n> | WMEMory<n> | FUNction<n> | FFT | DIGital<n>}
 <n> ::= an integer in NR1 format.

When an annotation's mode is SOURce (see :DISPlay:ANNotation<n>:MODE), the :DISPlay:ANNotation<n>:SOURce command makes the association to a waveform source.

Query Syntax :DISPlay:ANNotation<n>:SOURce?

The :DISPlay:ANNotation<n>:SOURce? query returns the waveform source to which the annotation is associated.

Return Format <source><NL>
 <source> ::= {CHAN<n> | WMEM<n> | FUNC<n> | FFT | DIG<n>}

- See Also**
- [":DISPlay:ANNotation<n>:MODE"](#) on page 414
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:TEXT

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:TEXT <string>

<string> ::= quoted ASCII string (up to 254 characters)

<n> ::= an integer from 1 to 20 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 410
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:ANNotation<n>:XPOSition

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:XPOSition <value>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.

<n> ::= an integer from 1 to 20 in NR1 format.

The :DISPlay:ANNotation<n>:XPOSition command sets the annotation's horizontal X position.

Query Syntax :DISPlay:ANNotation<n>:XPOSition?

The :DISPlay:ANNotation<n>:XPOSition? query returns the annotation's horizontal X position.

Return Format <value><NL>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.

- See Also**
- [":DISPlay:ANNotation<n>:YPOSition"](#) on page 418
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416

:DISPlay:ANNotation<n>:YPOSition

N (see [page 1708](#))

Command Syntax :DISPlay:ANNotation<n>:YPOSition <value>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.

<n> ::= an integer from 1 to 20 in NR1 format.

The :DISPlay:ANNotation<n>:YPOSition command sets the annotation's vertical Y position.

Query Syntax :DISPlay:ANNotation<n>:YPOSition?

The :DISPlay:ANNotation<n>:YPOSition? query returns the annotation's vertical Y position.

Return Format <value><NL>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.

- See Also**
- [":DISPlay:ANNotation<n>:XPOSition"](#) on page 417
 - [":DISPlay:ANNotation<n>"](#) on page 410
 - [":DISPlay:ANNotation<n>:COLor"](#) on page 412
 - [":DISPlay:ANNotation<n>:BACKground"](#) on page 411
 - [":DISPlay:ANNotation<n>:TEXT"](#) on page 416

:DISPlay:BACKlight

N (see [page 1708](#))

Command Syntax :DISPlay:BACKlight {{0 | OFF} | {1 | ON}}

The :DISPlay:BACKlight command turns the display's backlight off or on.

:DISPlay:CLEar

N (see [page 1708](#))

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 409

:DISPlay:CLOCK:IGRid

N (see [page 1708](#))

Command Syntax :DISPlay:CLOCK:IGRid {{0 | OFF} | {1 | ON}}

When the oscilloscope's clock information is displayed (see :DISPlay:CLOCK[:STATe]), the :DISPlay:CLOCK:IGRid command specifies whether the clock information is displayed in the first waveform grid (ON) or in the badges bar at the top of the screen (OFF).

Query Syntax :DISPlay:CLOCK:IGRid?

The :DISPlay:CLOCK:IGRid? query returns whether the oscilloscope's clock information is displayed in the first waveform grid (1) or in the badges bar at the top of the screen (0).

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:CLOCK:IGRid:XPOStion"](#) on page 422
 - [":DISPlay:CLOCK:IGRid:YPOStion"](#) on page 423
 - [":DISPlay:CLOCK\[:STATe\]"](#) on page 424

:DISPlay:CLOCK:IGRid:XPOStion

N (see [page 1708](#))

Command Syntax :DISPlay:CLOCK:IGRid:XPOStion <value>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.

When the oscilloscope's clock information is displayed (see :DISPlay:CLOCK[:STATe]) and is located in the first waveform grid (see :DISPlay:CLOCK:IGRid), the :DISPlay:CLOCK:IGRid:XPOStion command specifies the horizontal X position of the clock information as a percent of the grid width.

Query Syntax :DISPlay:CLOCK:IGRid:XPOStion?

The :DISPlay:CLOCK:IGRid:XPOStion? query returns the oscilloscope's clock information horizontal X position as a percent of the grid width.

Return Format <value><NL>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid width.

- See Also**
- [":DISPlay:CLOCK:IGRid"](#) on page 421
 - [":DISPlay:CLOCK:IGRid:YPOStion"](#) on page 423
 - [":DISPlay:CLOCK\[:STATe\]"](#) on page 424

:DISPlay:CLOCK:IGRid:YPOSition

N (see [page 1708](#))

Command Syntax :DISPlay:CLOCK:IGRid:YPOSition <value>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.

When the oscilloscope's clock information is displayed (see :DISPlay:CLOCK[:STATe]) and is located in the first waveform grid (see :DISPlay:CLOCK:IGRid), the :DISPlay:CLOCK:IGRid:YPOSition command specifies the vertical Y position of the clock information as a percent of the grid height.

Query Syntax :DISPlay:CLOCK:IGRid:YPOSition?

The :DISPlay:CLOCK:IGRid:YPOSition? query returns the oscilloscope's clock information vertical Y position as a percent of the grid height.

Return Format <value><NL>

<value> ::= a decimal from 0.0 to 1.0 in NR3 format that is a percent of the grid height.

- See Also**
- [":DISPlay:CLOCK:IGRid"](#) on page 421
 - [":DISPlay:CLOCK:IGRid:XPOSition"](#) on page 422
 - [":DISPlay:CLOCK\[:STATe\]"](#) on page 424

:DISPlay:CLOCK[:STATe]

N (see [page 1708](#))

Command Syntax :DISPlay:CLOCK[:STATe] {{0 | OFF} | {1 | ON}}

The :DISPlay:CLOCK[:STATe] command enables or disables the display of the oscilloscope's clock information.

Query Syntax :DISPlay:CLOCK[:STATe] ?

The :DISPlay:CLOCK[:STATe]? query returns whether the oscilloscope's clock information is displayed on screen.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:CLOCK:IGRId"](#) on page 421
 - [":DISPlay:CLOCK:IGRId:XPOSition"](#) on page 422
 - [":DISPlay:CLOCK:IGRId:YPOSition"](#) on page 423

:DISPlay:DATA?

N (see [page 1708](#))

Query Syntax :DISPlay:DATA? [<format>]

<format> ::= {BMP | PNG}

The :DISPlay:DATA? query reads screen image data. You can choose 24-bit BMP or 24-bit PNG formats.

If no format option is specified, the screen image is returned as specified by the front panel's **File > Save...** dialog box's **Format** selection.

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 409
 - [":HCOPY:SDUMp:DATA?"](#) on page 574
 - [":HCOPY:SDUMp\[:DATA\]:FORMat"](#) on page 575
 - ["*RCL \(Recall\)"](#) on page 246
 - ["*SAV \(Save\)"](#) on page 250
 - [":VIEW"](#) on page 281

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? PNG"
byteData = myScope.ReadIEEEBlock(BinaryType_UI1)
' Output display data to a file:
strPath = "c:\scope\data\screen.png"
' 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 49](#), "Programming Examples," starting on page 1717

:DISPlay:DIALog:CLOSe

N (see [page 1708](#))

Command Syntax :DISPlay:DIALog:CLOSe

The :DISPlay:DIALog:CLOSe command closes all open dialog boxes in the graphical user interface. You may want to send this command before saving screen images.

See Also • [":DISPlay:DIALog:TRANSPARENT"](#) on page 427

:DISPlay:DIALog:TRANSPARENT

N (see [page 1708](#))

Command Syntax :DISPlay:DIALog:TRANSPARENT {{0 | OFF} | {1 | ON}}

The :DISPlay:DIALog:TRANSPARENT command enables or disables the **Transparent Dialogs** user preference option. When enabled, dialog box backgrounds are transparent. When disabled, dialog box backgrounds are solid.

Query Syntax :DISPlay:DIALog:TRANSPARENT?

The :DISPlay:DIALog:TRANSPARENT? query returns the transparent dialog box background setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

See Also • [":DISPlay:DIALog:CLOSe"](#) on page 426

:DISPlay:GRATicule:ALABels

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:ALABels {{0 | OFF} | {1 | ON}}

The :DISPlay:GRATicule:ALABels command turns graticule (grid) axis labels on or off.

Query Syntax :DISPlay:GRATicule:ALABels?

The :DISPlay:GRATicule:ALABels? query returns the graticule (grid) axis labels setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:GRATicule:INTensity"](#) on page 433
 - [":DISPlay:GRATicule:ALABels:DUAL"](#) on page 429
 - [":DISPlay:GRATicule:ALABels:IGrid"](#) on page 430

:DISPlay:GRATicule:ALABels:DUAL

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:ALABels:DUAL {{0 | OFF} | {1 | ON}}

When multiple waveforms are displayed in a grid, grid scales are displayed (see :DISPlay:GRATicule:ALABels), and scales are not shown within the grid (see :DISPlay:GRATicule:ALABels:IGrid), the :DISPlay:GRATicule:ALABels:DUAL command specifies whether grid scales are shown for two waveforms.

Query Syntax :DISPlay:GRATicule:ALABels:DUAL?

The :DISPlay:GRATicule:ALABels:DUAL? query returns whether grid scales are shown for two waveforms.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:GRATicule:ALABels"](#) on page 428
 - [":DISPlay:GRATicule:ALABels:IGrid"](#) on page 430

:DISPlay:GRATicule:ALABels:IGRid

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:ALABels:IGRid {{0 | OFF} | {1 | ON}}

When grid scales are displayed (see :DISPlay:GRATicule:ALABels), the :DISPlay:GRATicule:ALABels:IGRid command specifies whether graticule (grid) axis labels are displayed in separate scale bars (OFF) or within the grid (ON).

In order to display grid axis scales for two waveforms (see :DISPlay:GRATicule:ALABels:DUAL), grid axis labels must be displayed in in separate scale bars.

Query Syntax :DISPlay:GRATicule:ALABels:IGRid?

The :DISPlay:GRATicule:ALABels:IGRid? query returns whether graticule (grid) axis labels are displayed in separate scale bars (0) or within the grid (1).

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:GRATicule:ALABels"](#) on page 428
 - [":DISPlay:GRATicule:ALABels:DUAL"](#) on page 429

:DISPlay:GRATicule:COUNT

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:COUNT <num_grids>

<value> ::= an integer from 1 to 4 in NR1 format.

When the graticule (grid) layout is set to CUSTom (see :DISPlay:GRATicule:LAYout), the :DISPlay:GRATicule:COUNT command specifies the number of waveform grids.

Query Syntax :DISPlay:GRATicule:COUNT?

The :DISPlay:GRATicule:COUNT? query returns the number of waveform grids.

Return Format <num_grids><NL>

<num_grids> ::= an integer from 1 to 4 in NR1 format.

- See Also**
- [":DISPlay:GRATicule:LAYout"](#) on page 434
 - [":DISPlay:GRATicule:SET"](#) on page 435
 - [":DISPlay:GRATicule:SOURce?"](#) on page 436

:DISPlay:GRATicule:FSCReen

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:FSCReen {{0 | OFF} | {1 | ON}}

The :DISPlay:GRATicule:FSCReen command specifies whether the waveform display grids occupy the full screen.

Query Syntax :DISPlay:GRATicule:FSCReen?

The :DISPlay:GRATicule:FSCReen? query returns the whether the waveform display grids occupy the full screen.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":DISPlay:RESults:SIZE"](#) on page 448
 - [":DISPlay:GRATicule:LAYout"](#) on page 434

:DISPlay:GRATicule:INTensity

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:INTensity <value>

<value> ::= an integer from 0 to 100 in NR1 format.

The :DISPlay:GRATicule:INTensity command sets the graticule (grid) intensity.

Query Syntax :DISPlay:GRATicule:INTensity?

The :DISPlay:GRATicule:INTensity? query returns the graticule (grid) intensity setting.

Return Format <value><NL>

See Also • [":DISPlay:GRATicule:ALABels"](#) on page 428

:DISPlay:GRATicule:LAYout

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:LAYout <type>

<type> ::= {STACked | TILed | OVERlay | CUSTom}

The :DISPlay:GRATicule:LAYout command specifies the layout of waveform display grids:

- **STACked** – Waveform display grids are stacked vertically.
Up to four grids are automatically used depending on the number of waveforms that are displayed. Assignment of waveforms to grids also happens automatically.
- **TILed** – Waveform display grids are tiled horizontally and vertically.
Up to four grids are automatically used depending on the number of waveforms that are displayed. Assignment of waveforms to grids also happens automatically.
- **OVERlay** – All waveforms are overlaid in one grid.
- **CUSTom** – Specifies a custom waveform display grid layout where the number of grids can be specified (see :DISPlay:GRATicule:COUNT) and waveform sources can be assigned to particular grids (see :DISPlay:GRATicule:SET).

Query Syntax :DISPlay:GRATicule:LAYout?

The :DISPlay:GRATicule:LAYout? query returns the waveform display grid layout.

Return Format <type><NL>

<type> ::= {STAC | TIL | OVER | CUST}

- See Also**
- [":DISPlay:GRATicule:COUNT"](#) on page 431
 - [":DISPlay:GRATicule:SET"](#) on page 435
 - [":DISPlay:GRATicule:SOURce?"](#) on page 436

:DISPlay:GRATicule:SET

N (see [page 1708](#))

Command Syntax :DISPlay:GRATicule:SET <source>,<grid>

<source> ::= {CHANnel<n> | WMEMory<n> | FUNction<n> | FFT
 | {PBUS<n> | SBUS<n>} | GAIN | PHASe}

<grid> ::= a grid number inter from 1 to 4 in NR1 format.

When the graticule (grid) layout is set to CUSTom (see :DISPlay:GRATicule:LAYout), the :DISPlay:GRATicule:SET command assigns a waveform source to a particular grid.

The specified grid must be enabled before a waveform source can be assigned to it (see :DISPlay:GRATicule:COUNt).

To view the waveform sources assigned to a grid, use the :DISPlay:GRATicule:SOURce? query.

- See Also**
- [":DISPlay:GRATicule:COUNt"](#) on page 431
 - [":DISPlay:GRATicule:LAYout"](#) on page 434
 - [":DISPlay:GRATicule:SOURce?"](#) on page 436

:DISPlay:GRATicule:SOURce?

N (see [page 1708](#))

Query Syntax :DISPlay:GRATicule:SOURce? <grid>

<grid> ::= a grid number integer from 1 to 4 in NR1 format.

The :DISPlay:GRATicule:SOURce? query returns a comma-separated list of the waveforms in the grid.

The list can contain items like: CHAN<n>, WMEM<n>, FUNC<n>, FFT, SBUS<n>, GAIN, PHASe

When the graticule (grid) layout is set to CUSTom (see :DISPlay:GRATicule:LAYout), you can assign waveform sources to a particular grid using the :DISPlay:GRATicule:SET command.

Return Format <sources><NL>

<sources> ::= comma-separated list of the waveforms displayed in the grid.

- See Also**
- [":DISPlay:GRATicule:COUNt"](#) on page 431
 - [":DISPlay:GRATicule:LAYout"](#) on page 434
 - [":DISPlay:GRATicule:SET"](#) on page 435

:DISPlay:INTensity:WAVeform

 (see [page 1708](#))

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 409

:DISPlay:LABel

N (see [page 1708](#))

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 409
 - [":CHANnel<n>:LABel"](#) on page 354

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 49](#), "Programming Examples," starting on page 1717

:DISPlay:LABList

N (see [page 1708](#))

Command Syntax :DISPlay:LABList <binary block data>

<binary block> ::= an ordered list of up to 75 labels, a maximum of 32 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 "A1234567890123456789012345678901", 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 32 characters each, separated by newline characters.

- See Also**
- ["Introduction to :DISPlay Commands"](#) on page 409
 - [":DISPlay:LABel"](#) on page 438
 - [":CHANnel<n>:LABel"](#) on page 354
 - [":DIGital<d>:LABel"](#) on page 401
 - [":BUS<n>:LABel"](#) on page 320

:DISPlay:MESSage:CLEar

N (see [page 1708](#))

Command Syntax :DISPlay:MESSage:CLEar

The :DISPlay:MESSage:CLEar command removes all user messages that are currently on screen.

See Also • [":SYSTem:DSP"](#) on page 1427

:DISPlay:PERSistence

N (see [page 1708](#))

Command Syntax :DISPlay:PERSistence <value>
 <value> ::= {MINimum | INFinite | <time>}
 <time> ::= seconds in 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:PERSistence:CLEar command to erase points stored by persistence.

NOTE

Persistence is not available for math or FFT functions.

Query Syntax :DISPlay:PERSistence?

The :DISPlay:PERSistence? query returns the specified persistence setting.

Return Format <value><NL>
 <value> ::= {MIN | INF | <time>}

- See Also**
- ["Introduction to :DISPlay Commands"](#) on page 409
 - [":DISPlay:PERSistence:CLEar"](#) on page 442

:DISPlay:PERStence:CLEar

N (see [page 1708](#))

Command Syntax :DISPlay:PERStence:CLEar

The :DISPlay:PERStence:CLEar command erases all persistence data from the display, leaving the data from the last acquisition.

If the oscilloscope is running, the display will begin to accumulate waveform and persistence data again.

The :DISPlay:CLEar command clears all waveform data from the display, including the data from the last acquisition.

- See Also**
- [":DISPlay:PERStence"](#) on page 441
 - [":DISPlay:CLEar"](#) on page 420

:DISPlay:RESults:CATalog?

N (see [page 1708](#))

Query Syntax :DISPlay:RESults:CATalog?

The :DISPlay:RESults:CATalog? query returns a comma-separated list of the windows in the results area.

The list can contain items like: MEAS, MARK, HIST, DVM, COUN, LIST, MTES, FRAN

Return Format <windows><NL>

<windows> ::= comma-separated list of the results windows displayed from left to right.

- See Also**
- [":DISPlay:RESults:LAYout"](#) on page 444
 - [":DISPlay:RESults:SIZE"](#) on page 448

:DISPlay:RESults:LAYout

N (see [page 1708](#))

Command Syntax :DISPlay:RESults:LAYout <format>

<format> ::= {TAB | LIST}

The :DISPlay:RESults:LAYout command specifies the layout of windows in the results area:

- TAB – Windows in the results area are tabbed.
- LIST – Windows in the results area are listed next to each other and buttons at the bottom of the results area can be used to highlight a particular window.

Query Syntax :DISPlay:RESults:LAYout?

The :DISPlay:RESults:LAYout? query returns the specified results area layout.

Return Format <format><NL>

<format> ::= {TAB | LIST}

- See Also**
- [":DISPlay:RESults:CATalog?"](#) on page 443
 - [":DISPlay:RESults:SIZE"](#) on page 448
 - [":DISPlay:RESults:LAYout:LIST\[:SElect\]"](#) on page 445
 - [":DISPlay:RESults:LAYout:TAB:LEFT\[:SElect\]"](#) on page 446
 - [":DISPlay:RESults:LAYout:TAB:RIGHT\[:SElect\]"](#) on page 447

:DISPlay:RESults:LAYout:LIST[:SElect]

N (see [page 1708](#))

Command Syntax :DISPlay:RESults:LAYout:LIST[:SElect] <results_window>

<results_window> ::= {MEAS | MARK | HIST | DVM | COUN | LIST | MTES | FR
AN}

When windows in the results area are listed next to each other and buttons at the bottom of the results area can be used to highlight a particular window (see :DISPlay:RESults:LAYout), the :DISPlay:RESults:LAYout:LIST[:SElect] command lets you select and highlight a particular window.

Query Syntax :DISPlay:RESults:LAYout:LIST[:SElect]?

The :DISPlay:RESults:LAYout:LIST[:SElect]? query returns the selected and highlighted window when windows in the results area are listed next to each other.

Return Format <results_window><NL>

<results_window> ::= {MEAS | MARK | HIST | DVM | COUN | LIST | MTES | FR
AN}

- See Also**
- [":DISPlay:RESults:LAYout"](#) on page 444
 - [":DISPlay:RESults:LAYout:LIST\[:SElect\]"](#) on page 445
 - [":DISPlay:RESults:LAYout:TAB:LEFT\[:SElect\]"](#) on page 446
 - [":DISPlay:RESults:LAYout:TAB:RIGHT\[:SElect\]"](#) on page 447

:DISPlay:RESults:LAYout:TAB:LEFT[:SElect]

N (see [page 1708](#))

Command Syntax :DISPlay:RESults:LAYout:TAB:LEFT[:SElect] <results_window>

<results_window> ::= {MEAS | MARK | HIST | LIST | MTES | FRAN}

When windows in the results area are tabbed (see :DISPlay:RESults:LAYout), the :DISPlay:RESults:LAYout:TAB:LEFT[:SElect] command selects a particular window to display in the left pane.

Query Syntax :DISPlay:RESults:LAYout:TAB:LEFT[:SElect]?

The :DISPlay:RESults:LAYout:TAB:LEFT[:SElect]? query returns the left pane's selected window.

Return Format <results_window><NL>

<results_window> ::= {MEAS | MARK | HIST | LIST | MTES | FRAN}

- See Also**
- [":DISPlay:RESults:LAYout"](#) on page 444
 - [":DISPlay:RESults:LAYout:LIST\[:SElect\]"](#) on page 445
 - [":DISPlay:RESults:LAYout:TAB:LEFT\[:SElect\]"](#) on page 446
 - [":DISPlay:RESults:LAYout:TAB:RIGHT\[:SElect\]"](#) on page 447

:DISPlay:RESults:LAYout:TAB:RIGHT[:SElect]

N (see [page 1708](#))

Command Syntax :DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] <results_window>

<results_window> ::= {DVM | COUN}

When windows in the results area are tabbed (see :DISPlay:RESults:LAYout), the :DISPlay:RESults:LAYout:TAB:RIGHT[:SElect] command selects a particular window to display in the right pane.

Query Syntax :DISPlay:RESults:LAYout:TAB:RIGHT[:SElect]?

The :DISPlay:RESults:LAYout:TAB:RIGHT[:SElect]? query returns the right pane's selected window.

Return Format <results_window><NL>

<results_window> ::= {DVM | COUN}

- See Also**
- [":DISPlay:RESults:LAYout"](#) on page 444
 - [":DISPlay:RESults:LAYout:LIST\[:SElect\]"](#) on page 445
 - [":DISPlay:RESults:LAYout:TAB:LEFT\[:SElect\]"](#) on page 446
 - [":DISPlay:RESults:LAYout:TAB:RIGHT\[:SElect\]"](#) on page 447

:DISPlay:RESults:SIZE

N (see [page 1708](#))

Command Syntax :DISPlay:RESults:SIZE <size>

<size> ::= {CUSTom,<height> | FULL | HIDDEN}

<height> ::= a pixel height integer from 36 to 1024 in NR1 format.

The :DISPlay:RESults:SIZE command specifies the size of the results area:

- CUSTom,<height> – Lets you specify the height of the results area.
- FULL – The results area takes the full height of the display. A full height results area is 1024 pixels high.
- HIDDEN – The results area is minimized so that only the titles of the windows are visible. A hidden results area is 36 pixels high.

Query Syntax :DISPlay:RESults:SIZE?

The :DISPlay:RESults:SIZE? query returns the size of the results area.

Return Format <height><NL>

<height> ::= a pixel height integer from 36 to 1024 in NR1 format.

- See Also**
- [":DISPlay:RESults:CATalog?"](#) on page 443
 - [":DISPlay:RESults:LAYout"](#) on page 444
 - [":DISPlay:GRATicule:FSCreen"](#) on page 432

:DISPlay:TRANSPARENT

N (see [page 1708](#))

Command Syntax :DISPlay:TRANSPARENT <setting>

<setting> ::= {OFF | ON}

The :DISPlay:TRANSPARENT command enables or disables transparent mode for dialog box backgrounds in the front panel graphical user interface.

This command maps to the **Transparent Dialogs** option that appears in the graphical user interface User Options dialog box's Preferences tab (**Utilities > User Options...**).

Query Syntax :DISPlay:TRANSPARENT?

The :DISPlay:TRANSPARENT? query returns the transparent setting.

Return Format <setting><NL>

<setting> ::= {OFF | ON}

See Also • ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:VECTors

 (see [page 1708](#))

Command Syntax :DISPlay:VECTors <vectors>

<vectors> ::= {{1 | ON} | {0 | OFF}}

The :DISPlay:VECTors command turns vector display on or off.

When vectors are turned on, the oscilloscope displays lines connecting sampled data points.

You can turn off vectors to view just waveform data points.

Query Syntax :DISPlay:VECTors?

The :DISPlay:VECTors? query returns the vectors setting.

Return Format <vectors><NL>

<vectors> ::= {1 | 0}

See Also • ["Introduction to :DISPlay Commands"](#) on page 409

:DISPlay:XYColor

N (see [page 1708](#))

Command Syntax :DISPlay:XYColor <xy_color_scheme>

<xy_color_scheme> ::= {VIRidis | HOT | ABYSS | SKY | INFerno}

The :DISPlay:XYColor command specifies the color scheme (and persistence color) used for the XY horizontal time mode waveform.

Query Syntax :DISPlay:XYColor?

The :DISPlay:XYColor? query returns the selected XY color scheme.

Return Format <xy_color_scheme><NL>

<xy_color_scheme> ::= {VIR | HOT | ABYS | SKY | INF}

- See Also**
- [":TIMEbase:MODE"](#) on page 1472
 - [":DISPlay:PERSistence"](#) on page 441

17 :DVM Commands

These commands control the digital voltmeter (DVM) feature.

Table 99 :DVM Commands Summary

Command	Query	Options and Query Returns
:DVM:ARANge {{0 OFF} {1 ON}} (see page 454)	:DVM:ARANge? (see page 454)	{0 1}
n/a	:DVM:CURREnt? (see page 455)	<dvm_value> ::= floating-point number in NR3 format
:DVM:ENABle {{0 OFF} {1 ON}} (see page 456)	:DVM:ENABle? (see page 456)	{0 1}
:DVM:MODE <mode> (see page 457)	:DVM:MODE? (see page 457)	<dvm_mode> ::= {ACRMs DC DCRMs}
:DVM:SOURce <source> (see page 458)	:DVM:SOURce? (see page 458)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format

:DVM:ARANge

N (see [page 1708](#))

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 458
 - [":DVM:ENABLE"](#) on page 456
 - [":DVM:MODE"](#) on page 457

:DVM:CURRent?

N (see [page 1708](#))

Query Syntax :DVM:CURRent?

The :DVM:CURRent? query returns the displayed 3-digit DVM value based on the current mode.

NOTE

It can take up to a few seconds after DVM analysis is enabled before this query starts to produce good results, that is, results other than +9.9E+37. To wait for good values after DVM analysis is enabled, programs should loop until a value less than +9.9E+37 is returned.

Return Format <dvm_value><NL>

<dvm_value> ::= floating-point number in NR3 format

- See Also**
- [":DVM:SOURce"](#) on page 458
 - [":DVM:ENABle"](#) on page 456
 - [":DVM:MODE"](#) on page 457

:DVM:ENABLE

N (see [page 1708](#))

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 458
 - [":DVM:MODE"](#) on page 457
 - [":DVM:ARANge"](#) on page 454

:DVM:MODE

N (see [page 1708](#))

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 456
 - [":DVM:SOURce"](#) on page 458
 - [":DVM:ARANge"](#) on page 454
 - [":DVM:CURREnt?"](#) on page 455
 - [":TRIGger:MODE"](#) on page 1494

:DVM:SOURce

N (see [page 1708](#))

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 456
 - [":DVM:MODE"](#) on page 457
 - [":DVM:ARANge"](#) on page 454
 - [":DVM:CURREnt?"](#) on page 455

18 :EXternal Trigger Commands

Control the input characteristics of the external trigger input. See ["Introduction to :EXternal Trigger Commands"](#) on page 459.

Table 100 :EXternal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXternal:BWLimit <bwlimit> (see page 460)	:EXternal:BWLimit? (see page 460)	<bwlimit> ::= {0 OFF}
:EXternal:PROBe <attenuation> (see page 461)	:EXternal:PROBe? (see page 461)	<attenuation> ::= probe attenuation ratio in NR3 format
:EXternal:RANGe <range>[<suffix>] (see page 462)	:EXternal:RANGe? (see page 462)	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V mV}
:EXternal:UNITs <units> (see page 463)	:EXternal:UNITs? (see page 463)	<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 +1.0000000E+00
```

:EXternal:BWLimit

C (see [page 1708](#))

Command Syntax :EXternal:BWLimit <bwlimit>

<bwlimit> ::= {0 | OFF}

The :EXternal:BWLimit 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:BWLimit?

The :EXternal:BWLimit? query returns the current setting of the low-pass filter (always 0).

Return Format <bwlimit><NL>

<bwlimit> ::= 0

- See Also**
- ["Introduction to :EXternal Trigger Commands"](#) on page 459
 - ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:HFReject"](#) on page 1484

:EXternal:PROBe

C (see [page 1708](#))

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 can be set from 0.001 to 10000 in a 1-2-5 sequence. 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 459
 - [":EXternal:RANGe"](#) on page 462
 - ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":CHANnel<n>:PROBe"](#) on page 356

:EXternal:RANGe

C (see [page 1708](#))

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 459
 - [":EXternal:PROBe"](#) on page 461
 - ["Introduction to :TRIGger Commands"](#) on page 1479

:EXternal:UNITs

N (see [page 1708](#))

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 459
 - ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":EXternal:RANGe"](#) on page 462
 - [":EXternal:PROBe"](#) on page 461
 - [":CHANnel<n>:UNITs"](#) on page 376

19 :FFT Commands

Control the FFT function in the oscilloscope. See ["Introduction to :FFT Commands"](#) on page 467.

Table 101 :FFT Commands Summary

Command	Query	Options and Query Returns
:FFT:AVERAge:COUNT <count> (see page 468)	:FFT:AVERAge:COUNT? (see page 468)	<count> ::= an integer from 2 to 65536 in NR1 format.
n/a	:FFT:BSIZE? (see page 469)	<bin_size> ::= the FFT resolution as a bin size frequency width in NR3 format.
:FFT:CENTer <frequency> (see page 470)	:FFT:CENTer? (see page 470)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from -25 GHz to 25 GHz.
:FFT:CLear (see page 471)	n/a	n/a
:FFT:DETection:POINTs <pts_per_span> (see page 472)	:FFT:DETection:POINTs? (see page 472)	<pts_per_span> ::= an integer from 640 to 65536 in NR1 format.
:FFT:DETection:TYPE <detection> (see page 473)	:FFT:DETection:TYPE? (see page 473)	<detection> ::= {OFF SAMPlE PPOSitive PNEGative AVERAge NORMAl}
:FFT:DISPlay {{0 OFF} {1 ON}} (see page 474)	:FFT:DISPlay? (see page 474)	<s> ::= 1-6, in NR1 format. {0 1}
:FFT:DMODE <display_mode> (see page 475)	:FFT:DMODE? (see page 475)	<display_mode> ::= {NORMAl AVERAge MAXHold MINHold}
:FFT:FREQuency:START <frequency> (see page 476)	:FFT:FREQuency:START? (see page 476)	<frequency> ::= the start frequency in NR3 format.

Table 101 :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFT:FREQuency:STOP <frequency> (see page 477)	:FFT:FREQuency:STOP? (see page 477)	<frequency> ::= the stop frequency in NR3 format.
:FFT:GATE <gating> (see page 478)	:FFT:GATE? (see page 478)	<gating> ::= {NONE ZOOM MARKers}
:FFT:OFFSet <offset> (see page 480)	:FFT:OFFSet? (see page 480)	<offset> ::= the value at center screen in NR3 format.
:FFT:RANGe <range> (see page 481)	:FFT:RANGe? (see page 481)	<range> ::= the full-scale vertical axis value in NR3 format.
n/a	:FFT:RBWidth? (see page 482)	<resolution_bw> ::= the FFT resolution as a resolution bandwidth frequency in NR3 format.
:FFT:READout <type> (see page 483)	:FFT:READout? (see page 483)	<type> ::= {OFF SRATe BSIZE RBWidth}
:FFT:REFeRence <level> (see page 484)	:FFT:REFeRence? (see page 484)	<level> ::= the current reference level in NR3 format.
:FFT:SCALE <scale value>[<suffix>] (see page 485)	:FFT:SCALE? (see page 485)	<scale_value> ::= integer in NR1 format. <suffix> ::= dB
:FFT:SOURce <source> (see page 486)	:FFT:SOURce? (see page 486)	<source> ::= {CHANnel<n> FUNCTion<m> MATH<m>} <n> ::= 1 to (# analog channels) in NR1 format. <m> ::= 1 to (# math functions) in NR1 format
:FFT:SPAN (see page 487)	:FFT:SPAN? (see page 487)	 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.
n/a	:FFT:SRATe? (see page 488)	<sample_rate> ::= the FFT resolution as a sample rate frequency in NR3 format.
:FFT:VTYPe <units> (see page 489)	:FFT:VTYPe? (see page 489)	<units> ::= {DECibel VRMS}
:FFT:WINDow <window> (see page 490)	:FFT:WINDow? (see page 490)	<window> ::= {RECTangular HANNing FLATtop BHARRis BARTlett}

Table 101 :FFT Commands Summary (continued)

Command	Query	Options and Query Returns
:FFTMax:OFFSet <offset>[<suffix>] (see page 492)	:FFTMax:OFFSet? (see page 492)	<offset> ::= the value at center screen in NR3 format. <suffix> ::= {dB mdB udB}
:FFTMax:RESet (see page 493)	n/a	n/a
:FFTMax:SCALe <scale_value>[<suffix >] (see page 494)	:FFTMax:SCALe? (see page 494)	<scale_value> ::= floating-point value in NR3 format. <suffix> ::= {dB mdB udB}
:FFTMax:STATe {{0 OFF} {1 ON}} (see page 495)	:FFTMax:STATe? (see page 495)	<setting> ::= {0 1}

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;AVER:COUN 8
```

:FFT:AVERage:COUNT

N (see [page 1708](#))

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 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:STARt"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:BSIZE?

N (see [page 1708](#))

Query Syntax :FFT:BSIZE?

The :FFT:BSIZE? query returns the FFT's bin size value.

Return Format <bin_size><NL>

<bin_size> ::= the FFT resolution as a bin size frequency width in NR3 format.

- See Also**
- [":FFT:RBWidth?"](#) on page 482
 - [":FFT:READout"](#) on page 483
 - [":FFT:SRATE?"](#) on page 488

:FFT:CENTer

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALe"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPE"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:CLEAr

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:STARt"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:DETection:POINts

N (see [page 1708](#))

Command Syntax :FFT:DETection:POINts <pts_per_span>

<pts_per_span> ::= an integer from 640 to 65536 in NR1 format.

For FFT (Magnitude) functions, when FFT detectors are used (see :FFT:DETection:TYPE), the :FFT:DETection:POINts command specifies the maximum number of points that detectors should decimate to. This is also the number of buckets that sampled FFT points are grouped into before the selected detection type reduction (decimation) is applied.

Query Syntax :FFT:DETection:POINts?

The :FFT:DETection:POINts? query returns the specified maximum number of points that detectors should decimate to.

Return Format <pts_per_span><NL>

<pts_per_span> ::= an integer from 640 to 65536 in NR1 format.

See Also • [":FFT:DETection:TYPE"](#) on page 473

:FFT:DETection:TYPE

N (see [page 1708](#))

Command Syntax :FFT:DETection:TYPE <detection>

```
<detection> ::= {OFF | SAMPlE | PPOSitive | PNEGative | AVERage | NORMal
}
```

For FFT (Magnitude) functions, the :FFT:DETection:TYPE command lets you set the FFT detector decimation type.

Detectors give you a way of manipulating the acquired data to emphasize different features of the data. Detectors reduce (decimate) the number of FFT points to at most the number of specified detector points. In this reduction, sampled FFT points are bucketized, that is, split into a number of groups that equals the specified number of detector points. Then, the points in each bucket are reduced to a single point according to the selected detection type. The detector types are:

- OFF — No detector is used.
- SAMPlE — Takes the point nearest to the center of every bucket.
- PPOSitive (+ Peak) — Takes the most positive point in every bucket.
- PNEGative (- Peak) — Takes the most negative point in every bucket.
- AVERage — Takes the average of all points in every bucket.
- NORMal — Implements a rosenfell algorithm. This method picks either the minimum or maximum sample in every bucket depending on whether the data is monotonically increasing, decreasing, or varying.

When detectors are used, the FFT's output is decimated, and any analysis is performed on the reduced or detected data set.

Query Syntax :FFT:DETection:TYPE?

The :FFT:DETection:TYPE? query returns ...

Return Format <detection><NL>

```
<detection> ::= {OFF | SAMP | PPOS | PNEG | AVER | NORM}
```

See Also • [":FFT:DETection:POINts"](#) on page 472

:FFT:DISPlay

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:DMODE

N (see [page 1708](#))

Command Syntax :FFT:DMODE <display_mode>

<display_mode> ::= {NORMAL | AVERage | MAXHold | MINHold}

The :FFT:DMODE command selects one of these FFT waveform display modes:

- **NORMAL** – this is the FFT waveform without any averaging or hold functions applied. This is how FFT math function waveforms are displayed.
- **AVERage** – the FFT waveform is averaged the selected number of times. 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.

- **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.
- **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.

Query Syntax :FFT:DMODE?

The :FFT:DMODE? query returns the selected FFT display mode.

Return Format <display_mode><NL>

<display_mode> ::= {NORMAL | AVERage | MAXHold | MINHold}

See Also • [":FFT:READout"](#) on page 483

:FFT:FREQuency:START

N (see [page 1708](#))

Command Syntax :FFT:FREQuency:START <frequency>

<frequency> ::= the start frequency in NR3 format.

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.

Query Syntax :FFT:FREQuency:START?

The :FFT:FREQuency:START? query returns the current start frequency in Hertz.

Return Format <frequency><NL>

<frequency> ::= the start frequency in NR3 format.

- See Also**
- [":FFT:AVERage:COUNT"](#) on page 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPE"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:FREQuency:STOP

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEAr"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:GATE

N (see [page 1708](#))

Command Syntax `:FFT:GATE <gating>`
 `<gating> ::= {NONE | ZOOM | MARKers}`

The :FFT:GATE command specifies the input waveform data on which the FFT function is calculated:

- **NONE** – The FFT function is performed on input waveform data from the Main window and is displayed in the Main window.

If the Zoomed time base is displayed (that is, WINDow is selected for :TIMEbase:MODE), the Zoom window shows "zoomed in" data from the Main window.

- **ZOOM** – This option is available when the Zoom window is displayed. The FFT function is performed on input waveform data from the Zoom window and is displayed in the Zoom window only. The FFT function will have better resolution because less decimation is needed for the measurement record.
- **MARKers** – The FFT function is performed on the input waveform data between the horizontal markers and is displayed in the Main window. The FFT function will have better resolution because less decimation is needed for the measurement record.

If the Zoomed time base is displayed, the Zoom window shows "zoomed in" data from the Main window, possibly containing the improved-resolution FFT function waveform, depending on the location of the Zoom window.

Because this selection can affect the resolution of the FFT function waveform, it can also affect measurements made on the FFT function waveform, depending on the selected measurement window (see :MEASure:WINDow). When you improve the resolution of an FFT function using the ZOOM or MARKers options, you will likely want to match the measurement window to look at the improved-resolution data.

Query Syntax `:FFT:GATE?`

The :FFT:GATE? query returns the gate setting.

Return Format `<gating><NL>`
 `<gating> ::= {NONE | ZOOM | MARK}`

- See Also**
- [":TIMEbase:MODE"](#) on page 1472
 - [":MEASure:WINDow"](#) on page 765
 - [":FFT:VTYPE"](#) on page 489
 - [":FFT:WINDow"](#) on page 490
 - [":MARKer:X1Position"](#) on page 637

- **":MARKer:X2Position"** on page 640
- **"Introduction to :FFT Commands"** on page 467

:FFT:OFFSet

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEAr"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFeRence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:RANGe

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:REFeRence"](#) on page 484
 - [":FFT:SCALe"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPE"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:RBWidth?

N (see [page 1708](#))

Query Syntax **:FFT:RBWidth?**

The :FFT:RBWidth? query returns the FFT's resolution bandwidth value.

Return Format **<resolution_bw><NL>**

<resolution_bw> ::= the FFT resolution as a resolution bandwidth frequency in NR3 format.

- See Also**
- **":FFT:BSIZE?"** on page 469
 - **":FFT:READout"** on page 483
 - **":FFT:SRATE?"** on page 488

:FFT:READout

N (see [page 1708](#))

Command Syntax :FFT:READout <type>

<type> ::= {OFF | SRATe | BSIZE | RBWidth}

The :FFT:READout command specifies how the FFT resolution is displayed in the FFT spectrum plot:

- OFF – the FFT resolution is not displayed.
- SRATe – the sample rate is displayed. To return the FFT's sample rate value use :FFT:SRATe?.
- BSIZE – the bin size is displayed. To return the FFT's bin size value use :FFT:BSIZE?.
- RBWidth – the resolution bandwidth is displayed. To return the FFT's resolution bandwidth value use :FFT:RBWidth?.

Query Syntax :FFT:READout?

The :FFT:READout? query returns the selected FFT resolution display option.

Return Format <type><NL>

<type> ::= {OFF | SRAT | BSIZ | RBW}

- See Also**
- [":FFT:BSIZE?"](#) on page 469
 - [":FFT:RBWidth?"](#) on page 482
 - [":FFT:SRATe?"](#) on page 488

:FFT:REFeRence

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:SCALE

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFeRence"](#) on page 484
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:STARt"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:SOURce

N (see [page 1708](#))

Command Syntax :FFT:SOURce <offset>

 <source> ::= {CHANnel<n> | FUNction<m> | MATH<m>}

 <n> ::= 1 to (# analog channels) in NR1 format.

 <m> ::= 1 to (# math functions) in NR1 format

The :FFT:SOURce command selects the source for the FFT function.

NOTE

To allow similarity to the :FUNCTION<m>:SOURce1 command/query, you can also use :FFT:SOURce1 for this command and query.

Query Syntax :FFT:SOURce?

The :FFT:SOURce? query returns the current source1 for the FFT function.

Return Format <source><NL>

 <source> ::= {CHANnel<n> | FUNction<m> | MATH<m>}

 <n> ::= 1 to (# analog channels) in NR1 format.

 <m> ::= 1 to (# math functions) in NR1 format

- See Also**
- [":FFT:AVERage:COUNT"](#) on page 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFeRence"](#) on page 484
 - [":FFT:SCALe"](#) on page 485
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPe"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:SPAN

N (see [page 1708](#))

Command Syntax :FFT: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 <NL>

 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

- See Also**
- [":FFT:AVERage:COUNT"](#) on page 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFerence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:VTYPE"](#) on page 489
 - [":FFT:WINDow"](#) on page 490

:FFT:SRATe?

N (see [page 1708](#))

Query Syntax **:FFT:SRATe?**

The :FFT:SRATe? query returns the FFT's sample rate value.

Return Format <sample_rate><NL>

<sample_rate> ::= the FFT resolution as a sample rate frequency in NR3 format.

- See Also**
- [":FFT:BSIZE?"](#) on page 469
 - [":FFT:RBWidth?"](#) on page 482
 - [":FFT:READout"](#) on page 483

:FFT:VTYPE

N (see [page 1708](#))

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 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480
 - [":FFT:RANGe"](#) on page 481
 - [":FFT:REFeRence"](#) on page 484
 - [":FFT:SCALE"](#) on page 485
 - [":FFT:SOURce"](#) on page 486
 - [":FFT:SPAN"](#) on page 487
 - [":FFT:FREQuency:START"](#) on page 476
 - [":FFT:FREQuency:STOP"](#) on page 477
 - [":FFT:WINDow"](#) on page 490

:FFT:WINDow

N (see [page 1708](#))

Command Syntax :FFT:WINDow <window>

<window> ::= {RECTangular | HANNing | FLATtop | BHARris | BARTlett}

The :FFT:WINDow command allows the selection of 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).
- BARTlett – (triangular, with end points at zero) window is similar to the Hanning window in that it is good for making accurate frequency measurements, but its higher and wider secondary lobes make it not quite as good for resolving frequencies that are close together.

Query Syntax :FFT:WINDow?

The :FFT:WINDow? query returns the value of the window selected for the FFT function.

Return Format <window><NL>

<window> ::= {RECT | HANN | FLAT | BHAR | BART}

- See Also**
- [":FFT:AVERage:COUNT"](#) on page 468
 - [":FFT:CENTer"](#) on page 470
 - [":FFT:CLEar"](#) on page 471
 - [":FFT:DISPlay"](#) on page 474
 - [":FFT:OFFSet"](#) on page 480

- **":FFT:RANGe"** on page 481
- **":FFT:REFerence"** on page 484
- **":FFT:SCALe"** on page 485
- **":FFT:SOURce"** on page 486
- **":FFT:SPAN"** on page 487
- **":FFT:FREQuency:STARt"** on page 476
- **":FFT:FREQuency:STOP"** on page 477
- **":FFT:VTYPE"** on page 489

:FFTMax:OFFSet

N (see [page 1708](#))

Command Syntax :FFTMax:OFFSet <offset>[<suffix>]

<offset> ::= the value at center screen in NR3 format.

<suffix> ::= {dB | mdB | udB}

When the FFT Max function is enabled (see :FFTMax:STaTe), the :FFTMax:OFFSet command sets its vertical offset (center of grid value).

When <suffix> is not used, the units are assumed to be dB.

Query Syntax :FFTMax:OFFSet?

The :FFTMax:OFFSet? query returns the FFT Max function vertical offset setting.

Return Format <offset><NL>

<offset> ::= the value at center screen in NR3 format.

- See Also**
- [":FFTMax:RESet"](#) on page 493
 - [":FFTMax:SCALe"](#) on page 494
 - [":FFTMax:STaTe"](#) on page 495

:FFTMax:RESet

N (see [page 1708](#))

Command Syntax :FFTMax:RESet

When the FFT Max function is enabled (see :FFTMax:STATe), the :FFTMax:RESet command clears the number of waveforms evaluated.

- See Also**
- [":FFTMax:OFFSet"](#) on page 492
 - [":FFTMax:SCALe"](#) on page 494
 - [":FFTMax:STATe"](#) on page 495

:FFTMax:SCALE

N (see [page 1708](#))

Command Syntax :FFTMax:SCALE <scale_value>[<suffix>]

<scale_value> ::= floating-point value in NR3 format.

<suffix> ::= {dB | mdB | udB}

When the FFT Max function is enabled (see :FFTMax:STATe), the :FFTMax:SCALE command sets its vertical scale.

When <suffix> is not used, the units are assumed to be dB.

Query Syntax :FFTMax:SCALE?

The :FFTMax:SCALE? query returns the FFT Max function vertical scale setting.

Return Format <scale_value><NL>

<scale_value> ::= floating-point value in NR3 format.

- See Also**
- [":FFTMax:OFFSet"](#) on page 492
 - [":FFTMax:RESet"](#) on page 493
 - [":FFTMax:STATe"](#) on page 495

:FFTMax:STATe

N (see [page 1708](#))

Command Syntax :FFTMax:STATe {{0 | OFF} | {1 | ON}}

When the FFT function is on (see :FFT:DISPlay), the :FFTMax:STATe command enables or disables the FFT Max function.

The FFT Max function is a Max Hold function applied to the FFT function.

Query Syntax :FFTMax:STATe?

The :FFTMax:STATe? query returns the FFT Max function on/off state.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":FFTMax:OFFSet"](#) on page 492
 - [":FFTMax:RESet"](#) on page 493
 - [":FFTMax:SCALe"](#) on page 494
 - [":FFT:DISPlay"](#) on page 474

20 :FRANalysis Commands

Control oscilloscope functions associated with the Frequency Response Analysis (FRA) feature, which is available in oscilloscope models that have a license-enabled built-in waveform generator. See ["Introduction to :FRANalysis Commands"](#) on page 498.

Table 102 :FRANalysis Commands Summary

Command	Query	Options and Query Returns
n/a	:FRANalysis:DATA? [SWEep SINGle] (see page 500)	<binary_block> ::= comma-separated data with newlines at the end of each row
:FRANalysis:ENABle { {0 OFF} {1 ON} } (see page 501)	:FRANalysis:ENABle? (see page 501)	{0 1}
:FRANalysis:FREQuency:MODE <setting> (see page 502)	:FRANalysis:FREQuency:MODE? (see page 502)	<setting> ::= {SWEep SINGle}
:FRANalysis:FREQuency:SINGle <value>[suffix] (see page 503)	:FRANalysis:FREQuency:SINGle? (see page 503)	<value> ::= {20 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:FRANalysis:FREQuency:STARt <value>[suffix] (see page 504)	:FRANalysis:FREQuency:STARt? (see page 504)	<value> ::= {20 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:FRANalysis:FREQuency:STOP <value>[suffix] (see page 505)	:FRANalysis:FREQuency:STOP? (see page 505)	<value> ::= {100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:FRANalysis:PPDecade <value> (see page 506)	:FRANalysis:PPDecade? (see page 506)	<value> ::= {10 20 30 40 50 60 70 80 90 100}
:FRANalysis:RUN (see page 507)	n/a	n/a

Table 102 :FRANalysis Commands Summary (continued)

Command	Query	Options and Query Returns
:FRANalysis:SOURce:IN Put <source> (see page 508)	:FRANalysis:SOURce:IN Put? (see page 508)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:FRANalysis:SOURce:OU TPut <source> (see page 509)	:FRANalysis:SOURce:OU TPut? (see page 509)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:FRANalysis:SWEep:POI Nts <points> (see page 510)	:FRANalysis:SWEep:POI Nts? (see page 510)	<points> ::= integer from 1-1000 in NR1 format
:FRANalysis:TRACe <selection> (see page 511)	:FRANalysis:TRACe? (see page 511)	<selection> ::= {NONE ALL GAIN PHASe}[, {GAIN PHASe}]
:FRANalysis:WGEN:LOAD <impedance> (see page 512)	:FRANalysis:WGEN:LOAD ? (see page 512)	<impedance> ::= {ONEMeg FIFTy}
:FRANalysis:WGEN:VOLT age <amplitude>, [<range>] (see page 513)	:FRANalysis:WGEN:VOLT age? [<range>] (see page 513)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F20HZ F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ F20MHZ}
:FRANalysis:WGEN:VOLT age:PROFile {{0 OFF} {1 ON}} (see page 515)	:FRANalysis:WGEN:VOLT age:PROFile? (see page 515)	{0 1}

Introduction to :FRANalysis Commands

The FRANalysis subsystem controls the Frequency Response Analysis feature in the oscilloscope.

The Frequency Response Analysis (FRA) feature controls the built-in waveform generator to sweep a sine wave across a range of frequencies while measuring the input to and output from a device under test (DUT). At each frequency, gain (A) and phase are measured and plotted on a frequency response chart.

Reporting the Setup

Use :FRANalysis? to query setup information for the FRANalysis subsystem.

Return Format

The following is a sample response from the :FRANalysis? query. In this case, the query was issued following a *RST command.

```
:FRAN:SOUR:INP CHAN1;OUTP CHAN2;:FRAN:FREQ:STAR +100E+00;  
STOP +20.000000E+06;:FRAN:WGEN:VOLT +200.0E-03;LOAD FIFT
```

:FRANalysis:DATA?

N (see [page 1708](#))

Query Syntax :FRANalysis:DATA? [SWEep | SINGle]

The :FRANalysis:DATA? query returns the frequency response analysis data.

The data is returned in four comma-separated columns of data for each step in the sweep: Frequency (Hz), Amplitude (Vpp), Gain (dB), and Phase (°).

You can use the :FRANalysis:TRACe command to specify whether to include gain, phase, both gain and phase, or neither in the frequency response analysis results.

The SWEep or SINGle option specifies whether to get the data from a sweep or single-frequency analysis (see :FRANalysis:FREQuency:MODE). If this option is not specified, the data from the sweep analysis is returned by default.

Return Format <binary_block><NL>

<binary_block> ::= comma-separated data with newlines at the end of each row

- See Also**
- [":FRANalysis:ENABle"](#) on page 501
 - [":FRANalysis:FREQuency:MODE"](#) on page 502
 - [":FRANalysis:FREQuency:SINGle"](#) on page 503
 - [":FRANalysis:FREQuency:STARt"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:TRACe"](#) on page 511
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513

:FRANalysis:ENABLE

N (see [page 1708](#))

Command Syntax :FRANalysis:ENABLE <setting>

<setting> ::= {{0 | OFF} | {1 | ON}}

The :FRANalysis:ENABLE command turns the Frequency Response Analysis (FRA) feature on or off.

Query Syntax :FRANalysis:ENABLE?

The :FRANalysis:ENABLE? query returns a flag indicating whether the Frequency Response Analysis (FRA) feature is on or off.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:FREQuency:STARt"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513

:FRANalysis:FREQuency:MODE

N (see [page 1708](#))

Command Syntax :FRANalysis:FREQuency:MODE <setting>

<setting> ::= {SWEep | SINGle}

The :FRANalysis:FREQuency:MODE command lets you select between the normal swept frequency response analysis or analysis at a single frequency, which can be useful when debugging.

Query Syntax :FRANalysis:FREQuency:MODE?

The :FRANalysis:FREQuency:MODE? query returns the frequency mode setting.

Return Format <setting><NL>

<setting> ::= {SWEep | SINGle}

- See Also**
- [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:FREQuency:SINGle"](#) on page 503
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:FREQuency:SINGle

N (see [page 1708](#))

Command Syntax :FRANalysis:FREQuency:SINGle <value>[suffix]

<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
| 20000000}

[suffix] ::= {Hz | kHz | MHz}

The :FRANalysis:FREQuency:SINGle command sets the single frequency value. The frequency 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 :FRANalysis:FREQuency:SINGle?

The :FRANalysis:FREQuency:SINGle? query returns the single frequency setting.

Return Format <value><NL>

<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
| 20000000}

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABLE"](#) on page 501
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:FREQuency:MODE"](#) on page 502
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:FREQuency:START

N (see [page 1708](#))

Command Syntax :FRANalysis:FREQuency:START <value>[suffix]

<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}

[suffix] ::= {Hz | kHz | MHz}

The :FRANalysis:FREQuency:START command sets the frequency sweep start value. The frequency 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 :FRANalysis:FREQuency:START?

The :FRANalysis:FREQuency:START? query returns the frequency sweep start setting.

Return Format <value><NL>

<value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000}

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABle"](#) on page 501
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:FREQuency:MODE"](#) on page 502
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:FREQuency:STOP

N (see [page 1708](#))

Command Syntax :FRANalysis:FREQuency:STOP <value>[suffix]

<value> ::= {100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 20000000
}

[suffix] ::= {Hz | kHz | MHz}

The :FRANalysis:FREQuency:STOP command sets the frequency sweep stop value. The frequency 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 :FRANalysis:FREQuency:STOP?

The :FRANalysis:FREQuency:STOP? query returns the frequency sweep stop setting.

Return Format <value><NL>

<value> ::= {100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 20000000
}

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABle"](#) on page 501
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:FREQuency:MODE"](#) on page 502
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:PPDecade

N (see [page 1708](#))

Command Syntax :FRANalysis:PPDecade <value>

<value> ::= {10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100}

The :FRANalysis:PPDecade command specifies the number of points per decade in the frequency response analysis.

Query Syntax :FRANalysis:PPDecade?

The :FRANalysis:PPDecade? query returns the points per decade setting.

Return Format <value><NL>

- See Also**
- [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:FREQuency:STARt"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:RUN

N (see [page 1708](#))

Command Syntax :FRANalysis:RUN

The :FRANalysis:RUN command performs the Frequency Response Analysis. This analysis controls the built-in waveform generator to sweep a sine wave across a range of frequencies while measuring the input to and output from a device under test (DUT). At each frequency, gain (A) and phase are measured and plotted on a Bode frequency response chart.

The :FRANalysis:APPLY command is a valid compatible alias for the :FRANalysis:RUN command.

When the frequency response analysis completes, you can use the :FRANalysis:DATA? query to get four comma-separated columns of data for each step in the sweep: Frequency (Hz), Amplitude (Vpp), Gain (dB), and Phase (°).

You can use the :FRANalysis:TRACe command to specify whether to include gain, phase, both gain and phase, or neither in the frequency response analysis results.

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**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABle"](#) on page 501
 - [":FRANalysis:FREQuency:MODE"](#) on page 502
 - [":FRANalysis:FREQuency:SINGle"](#) on page 503
 - [":FRANalysis:FREQuency:STARt"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:TRACe"](#) on page 511
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513
 - ["*ESR? \(Standard Event Status Register\)"](#) on page 239

:FRANalysis:SOURce:INPut

N (see [page 1708](#))

Command Syntax :FRANalysis:SOURce:INPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :FRANalysis:SOURce:INPut command specifies the analog input channel that is probing the input voltage to the device under test (DUT) in the frequency response analysis.

Query Syntax :FRANalysis:SOURce:INPut?

The :FRANalysis:SOURce:INPut? query returns the currently selected channel probing the input voltage.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABle"](#) on page 501
 - [":FRANalysis:FREQuency:STARt"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513

:FRANalysis:SOURce:OUTPut

N (see [page 1708](#))

Command Syntax :FRANalysis:SOURce:OUTPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :FRANalysis:SOURce:OUTPut command specifies the analog input channel that is probing the output voltage from the device under test (DUT) in the frequency response analysis.

Query Syntax :FRANalysis:SOURce:OUTPut?

The :FRANalysis:SOURce:OUTPut? query returns the currently selected channel probing the output voltage.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABLE"](#) on page 501
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:WGEN:LOAD"](#) on page 512
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513

:FRANalysis:SWEep:POINts

N (see [page 1708](#))

Command Syntax :FRANalysis:SWEep:POINts <points>

<points> ::= integer from 1-1000 in NR1 format

The :FRANalysis:SWEep:POINts command specifies the number of points in the frequency response analysis.

Depending on the number of points specified and the number of decades in the frequency range, this command can also change the :FRANalysis:PPDecade setting.

Query Syntax :FRANalysis:SWEep:POINts?

The :FRANalysis:SWEep:POINts? query returns the points setting.

Return Format <points><NL>

<points> ::= integer from 1-1000 in NR1 format

- See Also**
- [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515

:FRANalysis:TRACe

N (see [page 1708](#))

Command Syntax :FRANalysis:TRACe <selection>

<selection> ::= {NONE | ALL | GAIN | PHASe}[, {GAIN | PHASe}]

The :FRANalysis:TRACe command specifies whether to include gain, phase, both gain and phase, or neither in the frequency response analysis results.

NOTE

This command affects the oscilloscope's front panel graphical user interface (plot and table) as well as when saving analysis data.

Query Syntax :FRANalysis:TRACe?

The :FRANalysis:TRACe? query returns a comma-separated list of the types of data that are currently included in the frequency response analysis results, or "NONE" if neither gain nor phase data is included.

Return Format <selection_list><NL>

<selection_list> ::= {"NONE" | "GAIN" | "PHASe" | "GAIN, PHASe"}

- See Also**
- [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:DATA?"](#) on page 500

:FRANalysis:WGEN:LOAD

N (see [page 1708](#))

Command Syntax :FRANalysis:WGEN:LOAD <impedance>

<impedance> ::= {ONEMeg | FIFTy}

The :FRANalysis:WGEN: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 :FRANalysis:WGEN:LOAD?

The :FRANalysis:WGEN:LOAD? query returns the current expected output load impedance.

Return Format <impedance><NL>

<impedance> ::= {ONEM | FIFT}

- See Also**
- [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABLE"](#) on page 501
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:RUN"](#) on page 507
 - [":FRANalysis:SOURce:INPut"](#) on page 508
 - [":FRANalysis:SOURce:OUTPut"](#) on page 509
 - [":FRANalysis:WGEN:VOLTage"](#) on page 513

:FRANalysis:WGEN:VOLTage

N (see [page 1708](#))

Command Syntax :FRANalysis:WGEN:VOLTage <amplitude>,[<range>]

<amplitude> ::= amplitude in volts in NR3 format

<range> ::= {F10MILLIHZ | F100MILLIHZ | F1HZ | {F10HZ | F20HZ} | F100HZ
| F1KHZ | F10KHZ | F100KHZ | F1MHZ | F10MHZ | {F20MHZ | F100MHZ}}

The :FRANalysis:WGEN:VOLTage command specifies the waveform generator's output sine wave amplitude.

Use the :WGEN:VOLTage:OFFSet command to specify the offset voltage or DC level.

You can also specify the amplitude and offset indirectly using the :WGEN:VOLTage:HIGH and :WGEN: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.

When the amplitude profile setting is on (:FRANalysis:WGEN:VOLTage:PROFile ON), the <range> option specifies the initial ramp amplitude at the frequency setting. Amplitudes ramp between the settings specified for individual frequencies.

NOTE

The F20HZ and F20MHZ <range> options are allowed for backward compatibility; however, they map to the F10HZ and F100MHZ boundaries, respectively.

Query Syntax :FRANalysis:WGEN:VOLTage? [<range>]

The :FRANalysis:WGEN:VOLTage? query returns the currently specified waveform generator amplitude.

When the amplitude profile setting is on (:FRANalysis:WGEN:VOLTage:PROFile ON), the <range> option specifies the frequency whose initial ramp amplitude is returned.

Return Format <amplitude><NL>

<amplitude> ::= amplitude in volts in NR3 format

- See Also**
- [":FRANalysis:WGEN:VOLTage:PROFile"](#) on page 515
 - [":FRANalysis:DATA?"](#) on page 500
 - [":FRANalysis:ENABLE"](#) on page 501
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506

- **":FRANalysis:RUN"** on page 507
- **":FRANalysis:SOURce:INPut"** on page 508
- **":FRANalysis:SOURce:OUTPut"** on page 509
- **":FRANalysis:WGEN:LOAD"** on page 512

:FRANalysis:WGEN:VOLTage:PROFile

N (see [page 1708](#))

Command Syntax :FRANalysis:WGEN:VOLTage:PROFile {{0 | OFF} | {1 | ON}}

The :FRANalysis:WGEN:VOLTage:PROFile command enables or disables the ability to specify amplitude ramping within different decades.

Query Syntax :FRANalysis:WGEN:VOLTage:PROFile?

The :FRANalysis:WGEN:VOLTage:PROFile? query returns the voltage profile setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":FRANalysis:WGEN:VOLTage"](#) on page 513
 - [":FRANalysis:SWEep:POINts"](#) on page 510
 - [":FRANalysis:PPDecade"](#) on page 506
 - [":FRANalysis:FREQuency:START"](#) on page 504
 - [":FRANalysis:FREQuency:STOP"](#) on page 505

21 :FUNction<m> Commands

Control math functions in the oscilloscope. See "[Introduction to :FUNction<m> Commands](#)" on page 522.

Table 103 :FUNction<m> Commands Summary

Command	Query	Options and Query Returns
:FUNction<m>:AUTosetu p (see page 524)	n/a	n/a
:FUNction<m>:AUTosetu p:ENABle {{0 OFF} {1 ON}} (see page 525)	:FUNction<m>:AUTosetu p:ENABle? (see page 525)	<m> ::= 1 to (# math functions) in NR1 format <setting> ::= {0 1}
:FUNction<m>:AVERage: COUNT <count> (see page 526)	:FUNction<m>:AVERage: COUNT? (see page 526)	<count> ::= an integer from 2 to 65536 in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:BUS:CLOC k <source> (see page 527)	:FUNction<m>:BUS:CLOC k? (see page 527)	<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:SLOP e <slope> (see page 528)	:FUNction<m>:BUS:SLOP e? (see page 528)	<slope> ::= {NEGative POSitive EITHer} <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:BUS:YINC rement <value> (see page 529)	:FUNction<m>:BUS:YINC rement? (see page 529)	<value> ::= value per bus code, in NR3 format <m> ::= 1 to (# math functions) in NR1 format

Table 103 :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:BUS:YORigin <value> (see page 530)	:FUNCTION<m>:BUS:YORigin? (see page 530)	<value> ::= value at bus code = 0, in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:BUS:YUNits <units> (see page 531)	:FUNCTION<m>:BUS:YUNits? (see page 531)	<units> ::= {VOLT AMPere NONE} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:CLEar (see page 532)	n/a	n/a
:FUNCTION<m>:DISPlay {{0 OFF} {1 ON}} (see page 533)	:FUNCTION<m>:DISPlay? (see page 533)	{0 1} <m> ::= 1 to (# math functions) in NR1 format
n/a	:FUNCTION<m>[:FFT]:BSIZE? (see page 534)	<bin_size> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:CENTer <frequency> (see page 535)	:FUNCTION<m>[:FFT]:CENTer? (see page 535)	<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]:DETECTION:POINTS <number_of_buckets> (see page 536)	:FUNCTION<m>[:FFT]:DETECTION:POINTS? (see page 536)	<number_of_buckets> ::= an integer in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:DETECTION:TYPE <type> (see page 537)	:FUNCTION<m>[:FFT]:DETECTION:TYPE? (see page 537)	<type> ::= {OFF SAMPLE PPOSitive PNEGative NORMal AVERage} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:FREQUENCY:START <frequency> (see page 538)	:FUNCTION<m>[:FFT]:FREQUENCY:START? (see page 538)	<frequency> ::= the start frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>[:FFT]:FREQUENCY:STOP <frequency> (see page 539)	:FUNCTION<m>[:FFT]:FREQUENCY:STOP? (see page 539)	<frequency> ::= the stop frequency in NR3 format. <m> ::= 1 to (# math functions) in NR1 format

Table 103 :FUNction<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNction<m>:FFT:GATE <gating> (see page 540)	:FUNction<m>:FFT:GATE ? (see page 540)	<gating> ::= {NONE ZOOM MARKers} <m> ::= 1-4 in NR1 format
:FUNction<m>[:FFT]:PH ASe:REfErEnce <ref_point> (see page 542)	:FUNction<m>[:FFT]:PH ASe:REfErEnce? (see page 542)	<ref_point> ::= {TRIGger DISPlay} <m> ::= 1-4 in NR1 format
n/a	:FUNction<m>[:FFT]:RB Width? (see page 543)	<resolution_bw> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>[:FFT]:RE ADout<n> <readout_type> (see page 544)	:FUNction<m>[:FFT]:RE ADout<n>? (see page 544)	<readout_type> ::= {SRATe BSize RBWidth} <m> ::= 1 to (# math functions) in NR1 format <n> ::= 1-2 in NR1 format, 2 is for dedicated FFT function
:FUNction<m>[:FFT]:SP AN (see page 545)	:FUNction<m>[:FFT]:SP AN? (see page 545)	 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz. <m> ::= 1 to (# math functions) in NR1 format
n/a	:FUNction<m>[:FFT]:SR ATe? (see page 546)	<sample_rate> ::= Hz in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>[:FFT]:VT YPe <units> (see page 547)	:FUNction<m>[:FFT]:VT YPe? (see page 547)	<units> ::= {DECibel VRMS} for the FFT (magnitude) operation <units> ::= {DEGREes RADians} for the FFTPhase operation <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>[:FFT]:WI NDow <window> (see page 548)	:FUNction<m>[:FFT]:WI NDow? (see page 548)	<window> ::= {RECTangular HANNing FLATtop BHARris BARTlett} <m> ::= 1 to (# math functions) in NR1 format

Table 103 :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:FREQuency:BANDpass:CENTer <center_freq> (see page 549)	:FUNCTION<m>:FREQuency:BANDpass:CENTer? (see page 549)	<center_freq> ::= center frequency of band-pass filter in NR3 format
:FUNCTION<m>:FREQuency:BANDpass:WIDTh <freq_width> (see page 550)	:FUNCTION<m>:FREQuency:BANDpass:WIDTh? (see page 550)	<freq_width> ::= frequency width of band-pass filter in NR3 format
:FUNCTION<m>:FREQuency:HIGHPass <3dB_freq> (see page 551)	:FUNCTION<m>:FREQuency:HIGHPass? (see page 551)	<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 552)	:FUNCTION<m>:FREQuency:LOWPass? (see page 552)	<3dB_freq> ::= 3dB cutoff frequency value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:GATE <gating> (see page 553)	:FUNCTION<m>:GATE? (see page 554)	<gating> ::= {NONE ZOOM BOTH MARKers} <m> ::= 1-4 in NR1 format
:FUNCTION<m>:INTEgrate:IICondition {{0 OFF} {1 ON}} (see page 555)	:FUNCTION<m>:INTEgrate:IICondition? (see page 555)	{0 1} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:INTEgrate:IOFFset <input_offset> (see page 556)	:FUNCTION<m>:INTEgrate:IOFFset? (see page 556)	<input_offset> ::= DC offset correction in NR3 format. <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:LABel <string> (see page 557)	:FUNCTION<m>:LABel? (see page 557)	<string> ::= quoted ASCII string
:FUNCTION<m>:LINEar:GAIN <value> (see page 558)	:FUNCTION<m>:LINEar:GAIN? (see page 558)	<value> ::= 'A' in $Ax + B$, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:LINEar:OFFSet <value> (see page 559)	:FUNCTION<m>:LINEar:OFFSet? (see page 559)	<value> ::= 'B' in $Ax + B$, value in NR3 format <m> ::= 1 to (# math functions) in NR1 format

Table 103 :FUNction<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNction<m>:OFFSet <offset> (see page 560)	:FUNction<m>:OFFSet? (see page 560)	<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 page 561)	:FUNction<m>:OPERatio n? (see page 563)	<operation> ::= {ADD SUBTract MULTiply DIVide INTegrate DIFF FFT FFTPhase SQRT MAGNify ABSolute SQUare LN LOG EXP TEN LOWPass HIGHpass BANDpass AVERage LINear MAXimum MINimum PEAK MAXHold MINHold TREND} <m> ::= 1 to (# math functions) in NR1 format
:FUNction<m>:RANGe <range> (see page 565)	:FUNction<m>:RANGe? (see page 565)	<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 566)	:FUNction<m>:REFerenc e? (see page 566)	<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 103 :FUNCTION<m> Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCTION<m>:SCALE <scale value>[<suffix>] (see page 567)	:FUNCTION<m>:SCALE? (see page 567)	<scale value> ::= integer in NR1 format <suffix> ::= {V dB} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:SMOOTH:POINts <points> (see page 568)	:FUNCTION<m>:SMOOTH:POINts? (see page 568)	<points> ::= odd integer in NR1 format
:FUNCTION<m>:SOURCE1 <source> (see page 569)	:FUNCTION<m>:SOURCE1? (see page 569)	<source> ::= {CHANnel<n> FUNCTION<c> MATH<c> WMEMory<r> BUS} <n> ::= 1 to (# analog channels) in NR1 format <c> ::= {1 2 3}, must be lower than <m> <r> ::= 1 to (# ref waveforms) in NR1 format ::= {1 2} <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:SOURCE2 <source> (see page 571)	:FUNCTION<m>:SOURCE2? (see page 571)	<source> ::= {CHANnel<n> WMEMory<r> NONE} <n> ::= 1 to (# analog channels) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <m> ::= 1 to (# math functions) in NR1 format
:FUNCTION<m>:TREND:MEASUREMENT MEAS<n> (see page 572)	:FUNCTION<m>:TREND:MEASUREMENT? (see page 572)	<n> ::= # of installed measurement, from 1 to 8 <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 19**, “:FFT Commands,” starting on page 465.

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 561.

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 LIN;DISP 0;SOUR1 CHAN1;RANG +8.00E+00;OFFS +0.0E+00;  
LIN:GAIN +1.00E+00;OFFS +0.0E+00
```

:FUNCTION<m>:AUTosetup

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:AUTosetup

The :FUNCTION<m>:AUTosetup command automatically sets the Y scale and Y offset to optimum values for the math function waveform.

See Also • [":FUNCTION<m>:AUTosetup:ENABLE"](#) on page 525

:FUNCTION<m>:AUTosetup:ENABle

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:AUTosetup:ENABle {{0 | OFF} | {1 | ON}}

<m> ::= 1 to (# math functions) in NR1 format

The :FUNCTION<m>:AUTosetup:ENABle command lets you enable or disable the automatic setup of a math function's scale and offset.

- ON – The math function waveform scale and offset are automatically determined when the waveform is turned on, when a different operator is selected, or whenever you send :FUNCTION<m>:AUTosetup. The commands for manually changing these settings are unavailable. "Auto" is displayed in the math function badge, and mouse wheel scaling of values in the badge is unavailable.
- OFF – No automatic setup occurs unless you send :FUNCTION<m>:AUTosetup (which is always available). The commands for manually setting up the scale and offset are available, and their settings will not automatically change when the waveform is turned on (or off and then back on) or when a different operator is selected.

Query Syntax :FUNCTION<m>:AUTosetup:ENABle?

The :FUNCTION<m>:AUTosetup:ENABle? query returns whether the option is enabled or disabled.

Return Format <setting><NL>

<setting> ::= {0 | 1}

See Also • [":FUNCTION<m>:AUTosetup"](#) on page 524

:FUNCTION<m>:AVERage:COUNT

N (see [page 1708](#))

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 561

:FUNCTION<m>:BUS:CLOCK

N (see [page 1708](#))

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 561

:FUNCTION<m>:BUS:SLOPe

N (see [page 1708](#))

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 561

:FUNCTION<m>:BUS:YINCrement

N (see [page 1708](#))

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 561

:FUNCTION<m>:BUS:YORigin

N (see [page 1708](#))

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 561

:FUNCTION<m>:BUS:YUNits

N (see [page 1708](#))

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 561

:FUNCTION<m>:CLEar

N (see [page 1708](#))

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 526

:FUNction<m>:DISPlay

N (see [page 1708](#))

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 operates as specified by the other :FUNction<m> commands.

NOTE

Automatic vertical scaling of the math function waveform occurs when the function display is turned from off to on. If you wait for the operation to complete (with an *OPC? query for example), then query the math function's scale, you can see the vertical scaling that was automatically determined.

One math function can be displayed at a time. If one math function is on and then another math function is turned on, the first math function will be turned off.

When math functions are off, they are still calculated so that they can be cascaded, that is, used as a source for a higher math function.

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 522
 - ["*OPC \(Operation Complete\)"](#) on page 243
 - [":FUNction<m>:SCALE"](#) on page 567
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270
 - [":STATus?"](#) on page 278

:FUNCTION<m>[:FFT]:BSIZE?

N (see [page 1708](#))

Query Syntax :FUNCTION<m>[:FFT]:BSIZE?

<m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:BSIZE? query returns the Bin Size setting for the FFT.

Return Format <bin_size><NL>

<bin_size> ::= Hz in NR3 format

See Also • [":FUNCTION<m>\[:FFT\]:READout<n>"](#) on page 544

:FUNCTION<m>[:FFT]:CENTER

N (see [page 1708](#))

Command Syntax	<pre>:FUNCTION<m>[:FFT]:CENTER <frequency></pre> <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	<pre>:FUNCTION<m>[:FFT]:CENTER?</pre> The :FUNCTION<m>[:FFT]:CENTER? query returns the current center frequency in Hertz.
Return Format	<pre><frequency><NL></pre> <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 522
 - [":FUNCTION<m>\[:FFT\]:SPAN"](#) on page 545
 - [":TIMEbase\[:MAIN\]:RANGE"](#) on page 1470
 - [":TIMEbase\[:MAIN\]:SCALE"](#) on page 1471

:FUNCTION<m>[:FFT]:DETECTION:POINTS

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:DETECTION:POINTS <number_of_buckets>
 <number_of_buckets> ::= an integer from 640 to 64K in NR1 format
 <m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:DETECTION:POINTS command specifies the maximum number of points that the FFT detector should decimate to. This is also the number of buckets that sampled FFT points are grouped into before the selected detection type reduction (decimation) is applied.

The minimum number of points is 640.

The maximum number of points is the analysis record limit. The analysis record defaults to 64K points but its depth can be increased at the expense of waveform update rate (see :SYSTEM:PRECISION:LENGTH).

Query Syntax :FUNCTION<m>[:FFT]:DETECTION:POINTS?

The :FUNCTION<m>[:FFT]:DETECTION:POINTS? query returns the FFT detector points setting.

Return Format <number_of_buckets><NL>
 <number_of_buckets> ::= an integer in NR1 format

- See Also**
- [":FUNCTION<m>\[:FFT\]:DETECTION:TYPE"](#) on page 537
 - [":SYSTEM:PRECISION:LENGTH"](#) on page 1443

:FUNCTION<m>[:FFT]:DETECTION:TYPE

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:DETECTION:TYPE <type>

<type> ::= {OFF | SAMPLE | PPOSITIVE | PNEGATIVE | NORMAL | AVERAGE}

<m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:DETECTION:TYPE command sets the FFT detector decimation type.

Detectors give you a way of manipulating the acquired data to emphasize different features of the data. Detectors reduce (decimate) the number of FFT points to at most the number of specified detector points. In this reduction, sampled FFT points are bucketized, that is, split into a number of groups that equals the specified number of detector points. Then, the points in each bucket are reduced to a single point according to the selected detection type. The detector types are:

- OFF – No detector is used.
- SAMPLE – Takes the point nearest to the center of every bucket.
- PPOSITIVE (+ Peak) – Takes the most positive point in every bucket.
- PNEGATIVE (- Peak) – Takes the most negative point in every bucket.
- AVERAGE – Takes the average of all points in every bucket.
- NORMAL – Implements a rosenfell algorithm. This method to picks either the minimum or maximum sample in every bucket depending on whether the data is monotonically increasing, decreasing, or varying. For details, see the [Spectrum Analysis Basics application note](#) at www.keysight.com.

When detectors are used, the FFT's output is decimated, and any analysis is performed on the reduced or detected data set.

Query Syntax :FUNCTION<m>[:FFT]:DETECTION:TYPE?

The :FUNCTION<m>[:FFT]:DETECTION:TYPE? query returns the FFT detector type setting

Return Format <type><NL>

<type> ::= {OFF | SAMPLE | PPOSITIVE | PNEGATIVE | NORMAL | AVERAGE}

See Also • [":FUNCTION<m>\[:FFT\]:DETECTION:POINTS"](#) on page 536

:FUNCTION<m>[:FFT]:FREQUENCY:START

N (see [page 1708](#))

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 539
 - [":FUNCTION<m>\[:FFT\]:CENTER"](#) on page 535
 - [":FUNCTION<m>\[:FFT\]:SPAN"](#) on page 545

:FUNCTION<m>[:FFT]:FREQUENCY:STOP

N (see [page 1708](#))

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 538
 - [":FUNCTION<m>\[:FFT\]:CENTER"](#) on page 535
 - [":FUNCTION<m>\[:FFT\]:SPAN"](#) on page 545

:FUNCTION<m>:FFT:GATE

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:FFT:GATE <gating>

<m> ::= 1-4 in NR1 format

<gating> ::= {NONE | ZOOM | MARKers}

The :FUNCTION<m>:FFT:GATE command specifies the input waveform data on which the FFT math function is calculated:

- NONE – The FFT math function is performed on input waveform data from the Main window and is displayed in the Main window.

If the Zoomed time base is displayed (that is, WINDow is selected for :TIMEbase:MODE), the Zoom window shows "zoomed in" data from the Main window.

- ZOOM – This option is available when the Zoom window is displayed. The FFT math function is performed on input waveform data from the Zoom window and is displayed in the Zoom window only. The FFT math function will have better resolution because less decimation is needed for the measurement record.
- MARKers – The FFT math function is performed on the input waveform data between the horizontal markers and is displayed in the Main window. The FFT math function will have better resolution because less decimation is needed for the measurement record.

If the Zoomed time base is displayed, the Zoom window shows "zoomed in" data from the Main window, possibly containing the improved-resolution FFT math function waveform, depending on the location of the Zoom window.

Because this selection can affect the resolution of the FFT math function waveform, it can also affect measurements made on the FFT math function waveform, depending on the selected measurement window (see :MEASure:WINDow). When you improve the resolution of an FFT math function using the ZOOM or MARKers options, you will likely want to match the measurement window to look at the improved-resolution data.

Query Syntax :FUNCTION<m>:FFT:GATE?

The :FUNCTION<m>:FFT:GATE? query returns the gate setting.

Return Format <gating><NL>

<gating> ::= {NONE | ZOOM | MARK}

- See Also**
- [":TIMEbase:MODE"](#) on page 1472
 - [":MEASure:WINDow"](#) on page 765
 - [":FUNCTION<m>\[:FFT\]:VTYPE"](#) on page 547
 - [":FUNCTION<m>\[:FFT\]:WINDow"](#) on page 548

- **":FUNction<m>:OPERation"** on page 561
- **":MARKer:X1Position"** on page 637
- **":MARKer:X2Position"** on page 640
- **"Introduction to :FUNction<m> Commands"** on page 522

:FUNCTION<m>[:FFT]:PHASe:REFeRence

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:PHASe:REFeRence <ref_point>

<ref_point> ::= {TRIGger | DISPlay}

<m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:PHASe:REFeRence command sets the reference point for calculating the FFT Phase function to either the trigger point or beginning of the displayed waveform.

Query Syntax :FUNCTION<m>[:FFT]:PHASe:REFeRence?

The :FUNCTION<m>[:FFT]:PHASe:REFeRence? query returns the selected reference point.

Return Format <ref_point><NL>

<ref_point> ::= {TRIGger | DISPlay}

- See Also**
- [":FUNCTION<m>:OPERation"](#) on page 561
 - ["Introduction to :FUNCTION<m> Commands"](#) on page 522

:FUNCTION<m>[:FFT]:RBWidth?

N (see [page 1708](#))

Query Syntax :FUNCTION<m>[:FFT]:RBWidth?

<m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:RBWidth? query returns the Resolution Bandwidth setting for the FFT.

Return Format <resolution_bw><NL>

<resolution_bw> ::= Hz in NR3 format

See Also • [":FUNCTION<m>\[:FFT\]:READout<n>"](#) on page 544

:FUNCTION<m>[:FFT]:READout<n>

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:READout<n> <readout_type>

<readout_type> ::= {SRATe | BSIZe | RBWidth}

<m> ::= 1-4 in NR1 format

<n> ::= 1-2 in NR1 format, 2 is valid only on oscilloscopes that have the dedicated FFT function

The :FUNCTION<m>[:FFT]:READout<n> command selects from these types of readouts for the FFT:

- SRATe – Sample Rate
- BSIZe – Bin Size
- RBWidth – Resolution Bandwidth

Note that READout1 is used for both the dedicated FFT and the general-purpose math FFT function and READout2 is used only for oscilloscopes that have a dedicated FFT function (like the 3000T X-Series).

Query Syntax :FUNCTION<m>[:FFT]:READout<n>?

The :FUNCTION<m>[:FFT]:READout<n>? query returns the readout selection.

Return Format <readout_type><NL>

<readout_type> ::= {SRAT | BSIZ | RBW}

- See Also**
- [":FUNCTION<m>\[:FFT\]:BSIZe?"](#) on page 534
 - [":FUNCTION<m>\[:FFT\]:RBWidth?"](#) on page 543
 - [":FUNCTION<m>\[:FFT\]:SRATe?"](#) on page 546

:FUNCTION<m>[:FFT]:SPAN

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:SPAN

<m> ::= 1 to (# math functions) in NR1 format

 ::= 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 :FUNCTION<m>[:FFT]:SPAN command sets the frequency span of the display (left graticule to right graticule) when FFT (Fast Fourier Transform) is selected.

Query Syntax :FUNCTION<m>[:FFT]:SPAN?

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.

Return Format <NL>

 ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>\[:FFT\]:CENTer"](#) on page 535
 - [":TIMEbase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TIMEbase\[:MAIN\]:SCALE"](#) on page 1471

:FUNCTION<m>[:FFT]:SRATE?

N (see [page 1708](#))

Query Syntax :FUNCTION<m>[:FFT]:SRATE?

<m> ::= 1-4 in NR1 format

The :FUNCTION<m>[:FFT]:SRATE? query returns the Sample Rate setting for the FFT.

Return Format <sample_rate><NL>

<sample_rate> ::= Hz in NR3 format

See Also • [":FUNCTION<m>\[:FFT\]:READout<n>"](#) on page 544

:FUNCTION<m>[:FFT]:VTYPE

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:VTYPE <units>

<m> ::= 1 to (# math functions) in NR1 format

<units> ::= {DECibel | VRMS} for the FFT (magnitude) operation

<units> ::= {DEGREes | RADians} for the FFTPhase operation

The :FUNCTION<m>[:FFT]:VTYPE command specifies FFT vertical units.

For the FFT (Magnitude) operation units, DECibel equates to the user interface's Logarithmic selection, and VRMS equates to the user interface's Linear selection.

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} for the FFT (magnitude) operation

<units> ::= {DEGR | RAD} for the FFTPhase operation

- See Also**
- [":FUNCTION<m>:FFT:GATE"](#) on page 540
 - ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:OPERation"](#) on page 561

:FUNCTION<m>[:FFT]:WINDOW

N (see [page 1708](#))

Command Syntax :FUNCTION<m>[:FFT]:WINDOW <window>

<m> ::= 1 to (# math functions) in NR1 format

<window> ::= {RECTangular | HANNing | FLATtop | BHARris | BARTlett}

The :FUNCTION<m>[:FFT]:WINDOW command allows the selection of 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).
- BARTlett – (triangular, with end points at zero) window is similar to the Hanning window in that it is good for making accurate frequency measurements, but its higher and wider secondary lobes make it not quite as good for resolving frequencies that are close together.

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 | BART}

- See Also**
- [":FUNCTION<m>:FFT:GATE"](#) on page 540
 - ["Introduction to :FUNCTION<m> Commands"](#) on page 522

:FUNCTION<m>:FREQUENCY:BANDpass:CENTER

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:FREQUENCY:BANDpass:CENTER <center_freq>
 <center_freq> ::= center frequency of band-pass filter in NR3 format

The :FUNCTION<m>:FREQUENCY:BANDpass:CENTER command sets the center frequency of the band-pass filter.

Query Syntax :FUNCTION<m>:FREQUENCY:BANDpass:CENTER?

The :FUNCTION<m>:FREQUENCY:BANDpass:CENTER? query returns the band-pass filter's center frequency setting.

Return Format <center_freq><NL>

- See Also**
- [":FUNCTION<m>:FREQUENCY:BANDpass:WIDTH"](#) on page 550
 - [":FUNCTION<m>:OPERation"](#) on page 561

:FUNCTION<m>:FREQUENCY:BANDpass:WIDTH

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:FREQUENCY:BANDpass:WIDTH <freq_width>

<freq_width> ::= frequency width of band-pass filter in NR3 format

The :FUNCTION<m>:FREQUENCY:BANDpass:WIDTH command sets the frequency width of the band-pass filter. The width specifies the filter's -3 dB cutoff frequencies (center frequency minus half the width and center frequency plus half the width).

Query Syntax :FUNCTION<m>:FREQUENCY:BANDpass:WIDTH?

The :FUNCTION<m>:FREQUENCY:BANDpass:WIDTH? query returns the band-pass filter's frequency width setting.

Return Format <freq_width><NL>

- See Also**
- [":FUNCTION<m>:FREQUENCY:BANDpass:CENTer"](#) on page 549
 - [":FUNCTION<m>:OPERation"](#) on page 561

:FUNction<m>:FREQuency:HIGHpass

N (see [page 1708](#))

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 561

:FUNCTION<m>:FREQUENCY:LOWPass

N (see [page 1708](#))

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 561

:FUNCTION<m>:GATE

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:GATE <gating>

<m> ::= 1-4 in NR1 format

<gating> ::= {NONE | ZOOM | BOTH | MARKers}

The :FUNCTION<m>:GATE command specifies the input waveform data on which the math function is calculated:

- NONE – The math function is performed on input waveform data from the Main window and is displayed in the Main window.

If the Zoomed time base is displayed (that is, WINDow is selected for :TIMEbase:MODE), the Zoom window shows "zoomed in" data from the Main window.

- ZOOM – This option is available when the Zoom window is displayed. The math function is performed on input waveform data from the Zoom window and is displayed in the Zoom window only. The math function will have better resolution because less decimation is needed for the measurement record.
- BOTH – This option is available when the Zoom window is displayed. In the Main time base window, the math function is performed on input waveform data from the Main window and is displayed in the Main window. In the Zoomed time base window, the math function is performed on input waveform data from the Zoom window and is displayed in the Zoom window. In other words, separate math function waveforms are shown in each window (calculated from the data in the respective windows).

This is the default setting when the Zoom window is displayed.

NOTE

The BOTH option is not available with FFT or FFTPhase math functions (see :FUNCTION<m>:OPERation) because the FFT length is different in the Main and Zoom windows. However, you can set up two FFT math functions on the same source, setting the measurement window on one of them to MAIN, and setting the measurement window on the other one to ZOOM.

- MARKers – The math function is performed on the input waveform data between the horizontal markers and is displayed in the Main window. The math function will have better resolution because less decimation is needed for the measurement record.

If the Zoomed time base is displayed, the Zoom window shows "zoomed in" data from the Main window, possibly containing the improved-resolution math function waveform, depending on the location of the Zoom window.

Because this selection can affect the resolution of the math function waveform, it can also affect measurements made on the math function waveform, depending on the selected measurement window (see :MEASure:WINDow). When you improve the resolution of a math function using the ZOOM or MARKers options, you will likely want to match the measurement window to look at the improved-resolution data.

Query Syntax :FUNCTION<m>:GATE?

The :FUNCTION<m>:GATE? query returns the gate setting.

Return Format <gating><NL>

<gating> ::= {NONE | ZOOM | BOTH | MARK}

- See Also**
- [":TIMEbase:MODE"](#) on page 1472
 - [":MEASure:WINDow"](#) on page 765
 - [":FUNCTION<m>:OPERation"](#) on page 561
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640
 - ["Introduction to :FUNCTION<m> Commands"](#) on page 522

:FUNCTION<m>:INTEgrate:IICondition

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:INTEgrate:IICondition {{0 | OFF} | {1 | ON}}

<m> ::= 1 to (# math functions) in NR1 format

The :FUNCTION<m>:INTEgrate:IICondition command sets the Initial Condition of the Integrate math function waveform:

- ON – the integrate math waveform is vertically centered in the screen. In other words, the top and bottom of the math waveform are equal distances from the top and bottom of the screen.
- OFF – the integrate math waveform starts at the zero-level reference on the left side of the screen.

Query Syntax :FUNCTION<m>:INTEgrate:IICondition?

The :FUNCTION<m>:INTEgrate:IICondition? query returns Initial Condition setting for the Integrate math function.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":FUNCTION<m>:OPERation"](#) on page 561
 - [":FUNCTION<m>:INTEgrate:IOffset"](#) on page 556

:FUNCTION<m>:INTEgrate:IOfFset

N (see [page 1708](#))

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 522
 - [":FUNCTION<m>:OPERation"](#) on page 561
 - [":FUNCTION<m>:INTEgrate:IICondition"](#) on page 555

:FUNCTION<m>:LABEL

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:LABEL <string>
 <m> ::= 1 to (# math functions) in NR1 format
 <string> ::= quoted ASCII string

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.

The :FUNCTION<m>:LABEL command sets the math function label to the string that follows. Setting a label for a function also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

Query Syntax :FUNCTION<m>:LABEL?

The :FUNCTION<m>:LABEL? query returns the label associated with a particular math function.

Return Format <string><NL>
 <string> ::= quoted ASCII string

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:DISPLAY"](#) on page 533
 - [":DISPLAY:LABEL"](#) on page 438
 - [":DISPLAY:LABList"](#) on page 439
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270
 - [":STATUS?"](#) on page 278

:FUNCTION<m>:LINEar:GAIN

N (see [page 1708](#))

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 561

:FUNCTION<m>:LINear:OFFSet

N (see [page 1708](#))

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 561

:FUNCTION<m>:OFFSet

N (see [page 1708](#))

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 522
 - [":FUNCTION<m>:RANGe"](#) on page 565
 - [":FUNCTION<m>:REFeRence"](#) on page 566
 - [":FUNCTION<m>:SCALe"](#) on page 567

:FUNCTION<m>:OPERation

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:OPERation <operation>

<m> ::= 1 to (# math functions) in NR1 format

<operation> ::= {ADD | SUBtract | MULTiply | DIVide | DIFF | INTEgrate
| FFT | FFTPhase | SQRT | MAGNify | ABSolute | SQUare | LN | LOG
| EXP | TEN | LOWPass | HIGHpass | BANDpass | AVERage | SMOoth
| ENvelope | LINear | MAXimum | MINimum | PEAK | MAXHold | MINHold
| TREND}

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 :FUNCTION<m>:INTEgrate:IOFFset command lets you specify a DC offset correction factor. The :FUNCTION<m>:INTEgrate:IICondition command lets you sets the Initial Condition of the Integrate math function waveform.
- FFT (magnitude) – Using the Fast Fourier Transform (FFT), this operation displays the magnitudes of the frequency content that makes up the source waveform. The FFT takes the digitized time record of the specified source and transforms it to the frequency domain.

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 decibels or V RMS.

- FFTPhase – Using the Fast Fourier Transform (FFT), this operation shows the phase relationships of the frequency content that makes up the source waveform. The FFT takes the digitized time record of the specified source and transforms it to the frequency domain.

The SPAN, CENTer, VTYPe, and WINDow commands are used for FFT functions. When FFTPhase is selected, the horizontal cursors change from time to frequency (Hz), and the vertical cursors change from volts to degrees or radians.

- 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.
 - BANDpass – Band pass filter – The FREQUENCY:BANDpass:CENTER and FREQUENCY:BANDpass:WIDTH commands set the center frequency and width of the filter.
 - 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:NMEasurement command selects the measurement whose trend you want to measure.

NOTE

If a math function is on (see :FUNCTION<m>:DISPlay), changing the operator will cause an automatic vertical scaling of the new math function waveform. If you wait for the operation to complete (with an *OPC? query for example), then query the math function's scale, you can see the vertical scaling that was automatically determined.

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 | FFTP
| Sqrt | MAGN | ABS | SQU | LN | LOG | EXP | TEN | LOWP | HIGH
| BAND | AVER | SMO | ENV | LIN | MAX | MIN | PEAK | MAXH | MINH
| TREN}
```

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:DISPlay"](#) on page 533
 - ["*OPC \(Operation Complete\)"](#) on page 243
 - [":FUNCTION<m>:SOURce1"](#) on page 569
 - [":FUNCTION<m>:SOURce2"](#) on page 571
 - [":FUNCTION<m>:INTEgrate:IICondition"](#) on page 555
 - [":FUNCTION<m>:INTEgrate:IOFFset"](#) on page 556
 - [":FUNCTION<m>\[:FFT\]:SPAN"](#) on page 545
 - [":FUNCTION<m>\[:FFT\]:CENTer"](#) on page 535
 - [":FUNCTION<m>\[:FFT\]:PHASe:REFerence"](#) on page 542
 - [":FUNCTION<m>\[:FFT\]:VTYPE"](#) on page 547
 - [":FUNCTION<m>\[:FFT\]:WINDow"](#) on page 548
 - [":FUNCTION<m>:LINear:GAIN"](#) on page 558
 - [":FUNCTION<m>:LINear:OFFSet"](#) on page 559
 - [":FUNCTION<m>:FREQuency:BANDpass:CENTer"](#) on page 549
 - [":FUNCTION<m>:FREQuency:BANDpass:WIDTh"](#) on page 550
 - [":FUNCTION<m>:FREQuency:LOWPass"](#) on page 552
 - [":FUNCTION<m>:FREQuency:HIGHPass"](#) on page 551
 - [":FUNCTION<m>:AVERage:COUNt"](#) on page 526
 - [":FUNCTION<m>:SMOoth:POINTs"](#) on page 568
 - [":FUNCTION<m>:TREND:NMEasurement"](#) on page 572
 - [":FUNCTION<m>:BUS:YINCrement"](#) on page 529
 - [":FUNCTION<m>:BUS:YORigin"](#) on page 530
 - [":FUNCTION<m>:BUS:YUNits"](#) on page 531
 - [":FUNCTION<m>:BUS:CLOCK"](#) on page 527
 - [":FUNCTION<m>:BUS:SLOPe"](#) on page 528

:FUNCTION<m>:RANGe

N (see [page 1708](#))

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 522
 - [":FUNCTION<m>:SCALE"](#) on page 567

:FUNCTION<m>:REFERENCE

N (see [page 1708](#))

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 522
 - [":FUNCTION<m>:OFFSET"](#) on page 560
 - [":FUNCTION<m>:RANGE"](#) on page 565
 - [":FUNCTION<m>:SCALE"](#) on page 567

:FUNCTION<m>:SCALE

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:SCALE <scale value>[<suffix>]
 <m> ::= 1 to (# math functions) in NR1 format
 <scale value> ::= vertical units/div value in NR3 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.

NOTE

Automatic vertical scaling of the math function waveform occurs when the function display is turned from off to on (see :FUNCTION<m>:DISPlay) or, if the function is already on, when the operation is changed (see :FUNCTION<m>:OPERation). If you want to change the math function's vertical scaling, you should do it after the math function display is turned on or after the operation is changed.

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> ::= vertical units/div value in NR3 format

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:DISPlay"](#) on page 533
 - [":FUNCTION<m>:OPERation"](#) on page 561
 - [":FUNCTION<m>:RANGe"](#) on page 565

:FUNCTION<m>:SMOoth:POINts

N (see [page 1708](#))

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 three up to half of the 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 561
 - [":SYSTem:PRECision:LENGth"](#) on page 1443

:FUNCTION<m>:SOURCE1

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:SOURCE1 <value>

<m> ::= 1 to (# math functions) in NR1 format

<value> ::= {CHANnel<n> | FUNCTION<c> | MATH<c> | WMemory<r> | BUS}

<n> ::= 1 to (# analog channels) in NR1 format

<c> ::= {1 | 2 | 3}, must be lower than <m>

<r> ::= 1 to (# ref waveforms) in NR1 format

 ::= {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 522) 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.

When :FUNCTION<m>:OPERation is TRENd, the :FUNCTION<m>:SOURCE1 command reports error -221, "Settings conflict" because the TRENd function operates on a measurement and not a source waveform.

Query Syntax :FUNCTION<m>:SOURCE1?

The :FUNCTION<m>:SOURce1? query returns the current source1 for function operations.

When :FUNCTION<m>:OPERation is TRENd, the :FUNCTION<m>:SOURce1? query returns the source of the measurement.

Return Format <value><NL>

<value> ::= {CHAN<n> | FUNC<c> | MATH<c> | WMEM<r> | BUS}

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:OPERation"](#) on page 561

:FUNCTION<m>:SOURce2

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:SOURce2 <value>

<m> ::= 1 to (# math functions) in NR1 format

<value> ::= {CHANnel<n> | WMEMory<r> | NONE}

<n> ::= 1 to (# analog channels) in NR1 format

<r> ::= 1 to (# ref waveforms) 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).

When :FUNCTION<m>:OPERation is TREND, the :FUNCTION<m>:SOURce2 command reports error -221, "Settings conflict" because the TREND function operates on a measurement and not a source waveform.

Query Syntax :FUNCTION<m>:SOURce2?

The :FUNCTION<m>:SOURce2? query returns the currently specified second source for math operations.

When :FUNCTION<m>:OPERation is TREND, the :FUNCTION<m>:SOURce2? query returns the source of the measurement.

Return Format <value><NL>

<value> ::= {CHAN<n> | WMEM<r> | NONE}

- See Also**
- ["Introduction to :FUNCTION<m> Commands"](#) on page 522
 - [":FUNCTION<m>:OPERation"](#) on page 561
 - [":FUNCTION<m>:SOURce1"](#) on page 569

:FUNCTION<m>:TREND:NMEasurement

N (see [page 1708](#))

Command Syntax :FUNCTION<m>:TREND:NMEasurement MEAS<n>

<n> ::= # of installed measurement, from 1 to 8

<m> ::= 1 to (# math functions) in NR1 format

The :FUNCTION<m>:TREND:NMEasurement command selects the measurement whose trend is shown in the math waveform.

There are 8 locations (or slots) that installed measurements can occupy. The MEAS<n> parameter specifies the location of the measurement whose trend you want to analyze.

You can view the trend math function waveform for these installed measurements:

- :MEASure:VAverage, <interval> ::= CYCLE
- :MEASure:VRMS, <type> ::= AC (AC RMS)
- :MEASure:VRATio, <interval> ::= CYCLE
- :MEASure:PERiod
- :MEASure:FREQuency
- :MEASure:PWIDth
- :MEASure:NWIDth
- :MEASure:DUTYcycle
- :MEASure:NDUTy
- :MEASure:RISetime
- :MEASure:FALLtime

Query Syntax :FUNCTION<m>:TREND:NMEasurement?

The :FUNCTION<m>:TREND:NMEasurement? query returns the selected measurement.

If no measurements are installed, the :FUNCTION<m>:TREND:NMEasurement? query will return NONE.

Return Format MEAS<n><NL>

<n> ::= # of installed measurement, from 1 to 8

See Also • [":FUNCTION<m>:OPERation"](#) on page 561

22 :HCOPy Commands

Commands for getting screen image data.

Table 104 :HCOPy Commands Summary

Command	Query	Options and Query Returns
n/a	:HCOPy:SDUMp:DATA? (see page 574)	<display_data> ::= binary block data in IEEE-488.2 # format.
:HCOPy:SDUMp[:DATA]:F ORMat <format> (see page 575)	:HCOPy:SDUMp[:DATA]:F ORMat? (see page 575)	<format> ::= {BMP PNG}

:HCOpy:SDUMp:DATA?

N (see [page 1708](#))

Query Syntax :HCOpy:SDUMp:DATA? [<format>]

<format> ::= {PNG | BMP}

The :HCOpy:SDUMp:DATA? query reads and returns screen image data. You can choose 24-bit BMP or 24-bit PNG formats.

In addition to the <format> option of this query, the screen image data format can also be set by the :HCOpy:SDUMp:DATA:FORMat command or the image **Format** selected in the graphical user interface's Save dialog box (**File > Save...**).

If no <format> option is specified with this query, the screen image data is returned in the currently selected format. After a *RST (factory default) command, the PNG format is selected by default.

If the <format> option is specified with this query, the format setting will affect the graphical user interface's **Format** selection in the Save dialog box.

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**
- [":HCOpy:SDUMp\[:DATA\]:FORMat"](#) on page 575
 - [":DISPlay:DATA?"](#) on page 425

:HCOPY:SDUMp[:DATA]:FORMat

N (see [page 1708](#))

Command Syntax :HCOPY:SDUMp[:DATA]:FORMat <format>

<format> ::= {PNG | BMP}

The :HCOPY:SDUMp[:DATA]:FORMat command specifies the format for screen image data: 24-bit PNG or 24-bit BMP.

The :HCOPY:SDUMp[:DATA]:FORMat setting will persist when cycling power but will be reset to PNG after a *RST (factory default) command.

The :HCOPY:SDUMp[:DATA]:FORMat setting will affect the graphical user interface's **Format** selection in the Save dialog box (**File > Save...**).

Query Syntax :HCOPY:SDUMp[:DATA]:FORMat?

The :HCOPY:SDUMp[:DATA]:FORMat? query returns the specified screen image data format.

The screen image data format can be set by the :HCOPY:SDUMp[:DATA]:FORMat command or the image **Format** selected in the graphical user interface's Save dialog box.

Return Format <format><NL>

<format> ::= {PNG | BMP}

- See Also**
- [":HCOPY:SDUMp:DATA?"](#) on page 574
 - [":DISPlay:DATA?"](#) on page 425
 - ["*RST \(Reset\)"](#) on page 247

23 :HISTogram Commands

These commands control the histogram analysis feature.

The histogram analysis feature provides a statistical view of a waveform or a measurement's results.

Waveform histograms show the number of times a waveform crosses into (or hits) a row or column of a user-defined window. For vertical histograms, the window is divided into rows. For horizontal histograms, the window is divided into columns.

Measurement histograms show the distribution of a measurement's results similar to ordinary measurement statistics.

As waveforms are acquired and displayed, or as measurements are made, a counter database accumulates the total number of hits (or measurements), and the histogram data is displayed as a bar graph on the graticule.

When the oscilloscope is stopped, that counter database remains intact until the acquired data display is modified, usually horizontally or vertically. At that time, the database is reset and the last displayed acquisition is used to start a new database.

Table 105 :HISTogram Commands Summary

Command	Query	Options and Query Returns
:HISTogram:AXIS <axis> (see page 580)	:HISTogram:AXIS? (see page 580)	<axis> ::= {VERTical HORizontal}
:HISTogram:DISPlay { {0 OFF} {1 ON} } (see page 581)	:HISTogram:DISPlay? (see page 581)	{0 1}
:HISTogram:MEASurement MEAS<n> (see page 582)	:HISTogram:MEASurement? (see page 582)	<n> ::= # of installed measurement, from 1 to 10
:HISTogram:MODE <mode> (see page 583)	:HISTogram:MODE? (see page 583)	<mode> ::= {OFF WAVEform MEASurement}
:HISTogram:RESet (see page 584)	n/a	n/a

Table 105 :HISTogram Commands Summary (continued)

Command	Query	Options and Query Returns
:HISTogram:SIZE <size> (see page 585)	:HISTogram:SIZE? (see page 585)	<size> ::= 1.0 to 5.0 for vertical histograms, 1.0 to 4.0 for horizontal histograms, in NR3 format
:HISTogram:TYPE <type> (see page 586)	:HISTogram:TYPE? (see page 586)	<type> ::= {VERTical HORizontal MEASurement}
:HISTogram:WINDow:BLIMit <limit> (see page 587)	:HISTogram:WINDow:BLIMit? (see page 587)	<limit> ::= bottom value in NR3 format
:HISTogram:WINDow:LLIMit <limit> (see page 588)	:HISTogram:WINDow:LLIMit? (see page 588)	<limit> ::= left value in NR3 format
:HISTogram:WINDow:RLIMit <limit> (see page 589)	:HISTogram:WINDow:RLIMit? (see page 589)	<limit> ::= right value in NR3 format
:HISTogram:WINDow:SOURce <source> (see page 590)	:HISTogram:WINDow:SOURce? (see page 590)	<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
:HISTogram:WINDow:TLMit <limit> (see page 591)	:HISTogram:WINDow:TLMit? (see page 591)	<limit> ::= top value in NR3 format

Histogram results statistics can be queried with these commands:

- **":MEASure:HISTogram:HITS?"** on page 695 – The sum of all hits in all bins (buckets) in the histogram.
- **":MEASure:HISTogram:PEAK?"** on page 704 – Maximum number of hits in any single bin.
- **":MEASure:HISTogram:MAXimum?"** on page 699 – The value corresponding to the maximum bin that has any hits.
- **":MEASure:HISTogram:MINimum?"** on page 702 – The value corresponding to the minimum bin that has any hits.
- **":MEASure:HISTogram:PPEak?"** on page 705 – Delta between the Max and Min values.

- **":MEASure:HISTogram:MEAN?"** on page 700 – Mean value of the histogram.
- **":MEASure:HISTogram:MEDian?"** on page 701 – Median value of the histogram.
- **":MEASure:HISTogram:MODE?"** on page 703 – Mode value of the histogram.
- **":MEASure:HISTogram:BWIDth?"** on page 694 – The width of each bin (bucket) in the histogram.
- **":MEASure:HISTogram:SDEViation?"** on page 706 – Standard deviation of the histogram values.
- **":MEASure:HISTogram:M1S?"** on page 696 – Percentage of histogram hits that lie within ± 1 Std Dev of the mean.
- **":MEASure:HISTogram:M2S?"** on page 697 – Percentage of histogram hits that lie within ± 2 Std Dev of the mean.
- **":MEASure:HISTogram:M3S?"** on page 698 – Percentage of histogram hits that lie within ± 3 Std Dev of the mean.

Histogram Interaction With Other Oscilloscope Features

When you enable a histogram, horizontal zoom, XY time mode, or roll time mode are automatically disabled.

When using segmented memory acquisitions, histograms perform similarly to math waveforms and measurements. The histogram is reset at the start of a segmented acquisition and all segments are histogrammed at the end of the acquisition.

Because it is not possible to histogram every waveform, be aware that persistence can show a peak or glitch that does not show up in the histogram data.

Histograms do not wait for the specified number of averaged acquisitions to be acquired before starting analysis, so intermediate results could be erratic.

:HISTogram:AXIS

N (see [page 1708](#))

Command Syntax :HISTogram:AXIS <axis>
 <axis> ::= {VERTical | HORizontal}

When the :HISTogram:MODE is WAVEform, the :HISTogram:AXIS command specifies the orientation of the histogram:

- HORizontal – the limits window is divided into columns, and the number of hits in each column is displayed in a histogram bar graph at the bottom of the graticule.
- VERTical – the limits window is divided into rows, and the number of hits in each row is displayed in a histogram bar graph at the left of the graticule.

(When the :HISTogram:MODE is MEASurement, histograms are always displayed horizontally at the bottom of the graticule.)

Query Syntax :HISTogram:AXIS?

The :HISTogram:AXIS? query returns the currently specified type of waveform histogram.

Return Format <axis><NL>
 <axis> ::= {VERT | HOR}

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:DISPlay

N (see [page 1708](#))

Command Syntax :HISTogram:DISPlay {{0 | OFF} | {1 | ON}}

The :HISTogram:DISPlay command turns the histogram display on or off.

Query Syntax :HISTogram:DISPlay?

The :HISTogram:DISPlay? query returns the current display setting for the histogram.

Return Format <off_on><NL>

{0 | 1}

- See Also**
- [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MEASurement"](#) on page 582
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:MEASurement

N (see [page 1708](#))

Command Syntax :HISTogram:MEASurement MEAS<n>

<n> ::= # of installed measurement, from 1 to 10

When the :HISTogram:MODE is MEASurement, the :HISTogram:MEASurement command selects the measurement to analyze.

There are 10 locations (or slots) that installed measurements can occupy. The MEAS<n> parameter specifies the location of the measurement you want to analyze.

Query Syntax :HISTogram:MEASurement?

The :HISTogram:MEASurement? query returns the location of the measurement currently selected for histogram analysis.

Return Format MEAS<n><NL>

<n> ::= # of installed measurement, from 1 to 10

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585

:HISTogram:MODE

N (see [page 1708](#))

Command Syntax :HISTogram:MODE <mode>
 <mode> ::= {OFF | WAVEform | MEASurement}

The :HISTogram:MODE command selects the type of histogram or disables the histogram analysis feature:

- OFF – disables the histogram analysis feature.
- WAVEform – shows the number of times a waveform crosses into (or hits) a row or column of a user-defined window. For vertical histograms, the window is divided into rows. For horizontal histograms, the window is divided into columns. Use the :HISTogram:AXIS command to specify a vertical or horizontal histogram.
- MEASurement – shows the distribution of a measurement's results just like ordinary measurement statistics.

Query Syntax :HISTogram:MODE?

The :HISTogram:MODE? query returns the currently set histogram mode.

Return Format <mode><NL>
 <mode> ::= {OFF | WAV | MEAS}

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MEASurement"](#) on page 582
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:RESet

N (see [page 1708](#))

Command Syntax :HISTogram:RESet

The :HISTogram:RESet command zeros the histogram counters.

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MEASurement"](#) on page 582
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:SIZE

N (see [page 1708](#))

Command Syntax :HISTogram:SIZE <size>

<size> ::= 1.0 to 5.0 for vertical histograms, 1.0 to 4.0 for horizontal histograms, in NR3 format

The :HISTogram:SIZE command specifies the number of divisions the histogram bar graph should use:

- For horizontal histograms, you can choose whole and half divisions from 1 to 4.
- For vertical histograms, you can choose whole and half divisions from 1 to 5.

Query Syntax :HISTogram:SIZE?

The :HISTogram:SIZE? query returns the number of divisions currently used for the histogram bar graph.

Return Format <size><NL>

<size> ::= 1.0 to 5.0 for vertical histograms, 1.0 to 4.0 for horizontal histograms, in NR3 format

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MEASurement"](#) on page 582
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:TYPE

 (see [page 1708](#))

Command Syntax :HISTogram:TYPE <type>
 <type> ::= {VERTical | HORizontal | MEASurement}

The :HISTogram:TYPE command selects either a waveform histogram (horizontal or vertical) or a measurement histogram:

- HORizontal – the limits window is divided into columns, and the number of hits in each column is displayed in a histogram bar graph at the bottom of the graticule.
- VERTical – the limits window is divided into rows, and the number of hits in each row is displayed in a histogram bar graph at the left of the graticule.
- MEASurement – shows the distribution of a measurement's results similar to ordinary measurement statistics.

Query Syntax :HISTogram:TYPE?

The :HISTogram:TYPE? query returns the currently selected histogram type.

Return Format <type><NL>
 <type> ::= {VERT | HOR | MEAS}

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MEASurement"](#) on page 582
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:WINDow:BLIMit

N (see [page 1708](#))

Command Syntax :HISTogram:WINDow:BLIMit <limit>

<limit> ::= bottom value in NR3 format

When the :HISTogram:MODE is WAVEform, the :HISTogram:WINDow:BLIMit command specifies the histogram limits window's bottom boundary.

The units are the same as the input waveform's vertical units. The input waveform is specified using the :HISTogram:WINDow:SOURce command.

Query Syntax :HISTogram:WINDow:BLIMit?

The :HISTogram:WINDow:BLIMit? query returns the histogram limits window's bottom boundary.

Return Format <limit><NL>

<limit> ::= bottom value in NR3 format

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:WINDow:LLIMit

N (see [page 1708](#))

Command Syntax :HISTogram:WINDow:LLIMit <limit>

<limit> ::= left value in NR3 format

When the :HISTogram:MODE is WAVEform, the :HISTogram:WINDow:LLIMit command specifies the histogram limits window's left boundary.

The units are the same as the input waveform's horizontal units. The input waveform is specified using the :HISTogram:WINDow:SOURce command.

Query Syntax :HISTogram:WINDow:LLIMit?

The :HISTogram:WINDow:LLIMit? query returns the histogram limits window's left boundary.

Return Format <limit><NL>

<limit> ::= left value in NR3 format

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:WINDow:RLIMit

N (see [page 1708](#))

Command Syntax :HISTogram:WINDow:RLIMit <limit>

<limit> ::= right value in NR3 format

When the :HISTogram:MODE is WAVEform, the :HISTogram:WINDow:RLIMit command specifies the histogram limits window's right boundary.

The units are the same as the input waveform's horizontal units. The input waveform is specified using the :HISTogram:WINDow:SOURce command.

Query Syntax :HISTogram:WINDow:RLIMit?

The :HISTogram:WINDow:RLIMit? query returns the histogram limits window's right boundary.

Return Format <limit><NL>

<limit> ::= right value in NR3 format

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:SOURce"](#) on page 590
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:WINDow:SOURce

N (see [page 1708](#))

Command Syntax :HISTogram:WINDow: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

When the :HISTogram:MODE is WAVEform, the :HISTogram:WINDow:SOURce command specifies the waveform that is used as the histogram source.

Query Syntax :HISTogram:WINDow:SOURce?

The :HISTogram:WINDow:SOURce? query returns the currently specified waveform histogram source.

Return Format <source><NL>

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:TLIMit"](#) on page 591

:HISTogram:WINDow:TLIMit

N (see [page 1708](#))

Command Syntax :HISTogram:WINDow:TLIMit <limit>

<limit> ::= top value in NR3 format

When the :HISTogram:MODE is WAVEform, the :HISTogram:WINDow:TLIMit command specifies the histogram limits window's top boundary.

The units are the same as the input waveform's vertical units. The input waveform is specified using the :HISTogram:WINDow:SOURce command.

Query Syntax :HISTogram:WINDow:TLIMit?

The :HISTogram:WINDow:TLIMit? query returns the histogram limits window's top boundary.

Return Format <limit><NL>

<limit> ::= top value in NR3 format

- See Also**
- [":HISTogram:DISPlay"](#) on page 581
 - [":HISTogram:TYPE"](#) on page 586
 - [":HISTogram:AXIS"](#) on page 580
 - [":HISTogram:MODE"](#) on page 583
 - [":HISTogram:RESet"](#) on page 584
 - [":HISTogram:SIZE"](#) on page 585
 - [":HISTogram:WINDow:BLIMit"](#) on page 587
 - [":HISTogram:WINDow:LLIMit"](#) on page 588
 - [":HISTogram:WINDow:RLIMit"](#) on page 589
 - [":HISTogram:WINDow:SOURce"](#) on page 590

24 :JITTer Commands

When the jitter analysis feature is licensed, these commands control jitter analysis.

Jitter analysis measures the variance of a measurement over time. The principal measurement used with jitter analysis is time interval error (TIE), but several other measurements can be analyzed as well.

The TIE measurement recovers an ideal clock frequency from the source waveform and compares it against measured transitions to generate error statistics. Several different clock recovery methods are available.

Jitter statistics can be visualized in four ways: measurement statistics, histogram, trend, and spectrum.

NOTE

It is important to set up absolute thresholds for the integrity of jitter measurements from acquisition to acquisition. See **":MEASure:THResholds:ABSolute"** on page 746.

Table 106 :JITTer Commands Summary

Command	Query	Options and Query Returns
:JITTer:HISTogram:APPLy (see page 595)	n/a	n/a
:JITTer:MEASurement MEAS<n> (see page 596)	:JITTer:MEASurement? (see page 596)	<n> ::= # of installed measurement, from 1 to 10
:JITTer:SPECTrum:APPLy (see page 598)	n/a	n/a
:JITTer:SPECTrum:HORizontal:POSition <value> (see page 599)	:JITTer:SPECTrum:HORizontal:POSition? (see page 599)	<value> ::= X position (center), in NR3 format
:JITTer:SPECTrum:HORizontal:RANGE <value> (see page 600)	:JITTer:SPECTrum:HORizontal:RANGE? (see page 600)	<value> ::= X scaling (span), in NR3 format

Table 106 :JITter Commands Summary (continued)

Command	Query	Options and Query Returns
:JITter:SPECTrum:VERTical:OFFSet <value> (see page 601)	:JITter:SPECTrum:VERTical:OFFSet? (see page 601)	<value> ::= Y offset, in NR3 format
:JITter:SPECTrum:VERTical:RANGe <value> (see page 602)	:JITter:SPECTrum:VERTical:RANGe? (see page 602)	<value> ::= Y scaling, in NR3 format
:JITter:SPECTrum:VERTical:TYPE <type> (see page 603)	:JITter:SPECTrum:VERTical:TYPE? (see page 603)	<type> ::= {DECibel VRMS}
:JITter:SPECTrum:WINDow <window> (see page 604)	:JITter:SPECTrum:WINDow? (see page 604)	<window> ::= {RECTangular HANNing FLATtop BHARRis BARTlett}
:JITter:STATe {{0 OFF} {1 ON}} (see page 606)	:JITter:STATe? (see page 606)	{0 1}
:JITter:TREND:APPLy (see page 608)	n/a	n/a
:JITter:TREND:SMOoth:APPLy (see page 609)	n/a	n/a
:JITter:TREND:SMOoth:POINTs <points> (see page 610)	:JITter:TREND:SMOoth:POINTs? (see page 610)	<points> ::= odd integer in NR1 format
:JITter:TREND:VERTical:OFFSet <value> (see page 611)	:JITter:TREND:VERTical:OFFSet? (see page 611)	<value> ::= Y offset, in NR3 format
:JITter:TREND:VERTical:RANGe <value> (see page 612)	:JITter:TREND:VERTical:RANGe? (see page 612)	<value> ::= Y scaling, in NR3 format

:JITTer:HISTogram:APPLy

N (see [page 1708](#))

Command Syntax :JITTer:HISTogram:APPLy

The :JITTer:HISTogram:APPLy command auto-configures and turns on the jitter measurement histogram for the selected measurement.

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:STaTe"](#) on page 606

:JITter:MEASurement

N (see [page 1708](#))

Command Syntax :JITter:MEASurement MEAS<n>

<n> ::= # of installed measurement, from 1 to 10

The :JITter:MEASurement command selects the measurement to analyze.

There are 10 locations (or slots) that installed measurements can occupy. The MEAS<n> parameter specifies the location of the measurement you want to analyze.

The measurement location you specify must have one of these measurements installed:

- Data TIE (time interval error) – see [":MEASure:TIEData"](#) on page 752.
Time interval error measurements use clock recovery. To change the clock recovery method, see [":CLOCK:METHod"](#) on page 382.
- Clock TIE (time interval error) – see [":MEASure:TIEClock"](#) on page 750.
Time interval error measurements use clock recovery. To change the clock recovery method, see [":CLOCK:METHod"](#) on page 382.
- N-Period – see [":MEASure:NPERiod"](#) on page 712.
- N-Period to N-Period – see [":MEASure:NCJitter"](#) on page 708.
- +Width to +Width – see [":MEASure:CTCPwidth"](#) on page 681.
- -Width to -Width – see [":MEASure:CTCNwidth"](#) on page 679.
- +Duty Cycle – see [":MEASure:DUTYcycle"](#) on page 687.

Query Syntax :JITter:MEASurement?

The :JITter:MEASurement? query returns the location of the measurement currently selected for histogram analysis.

Return Format MEAS<n><NL>

<n> ::= # of installed measurement, from 1 to 10

- See Also**
- [":JITter:HISTogram:APPLY"](#) on page 595
 - [":JITter:SPECTrum:APPLY"](#) on page 598
 - [":JITter:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITter:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITter:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITter:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITter:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITter:SPECTrum:WINDow"](#) on page 604

- **":JITTer:STATe"** on page 606
- **":JITTer:TRENd:APPLy"** on page 608
- **":JITTer:TRENd:SMOoth:APPLy"** on page 609
- **":JITTer:TRENd:SMOoth:POINts"** on page 610
- **":JITTer:TRENd:VERTical:OFFSet"** on page 611
- **":JITTer:TRENd:VERTical:RANGe"** on page 612

:JITTer:SPECTrum:APPLy

N (see [page 1708](#))

Command Syntax :JITTer:SPECTrum:APPLy

The :JITTer:SPECTrum:APPLy command auto-configures and turns on the jitter spectrum for the selected measurement.

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITTer:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITTer:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITTer:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITTer:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITTer:SPECTrum:WINDow"](#) on page 604
 - [":JITTer:STATe"](#) on page 606

:JITter:SPECTrum:HORizontal:POSition

N (see [page 1708](#))

Command Syntax :JITter:SPECTrum:HORizontal:POSition <value>

<value> ::= X position (center), in NR3 format

The :JITter:SPECTrum:HORizontal:POSition command sets the center frequency for the jitter measurement spectrum.

After the :JITter:SPECTrum:APPLY command, this command is an alias for :FUNCTION4[:FFT]:CENTER.

Query Syntax :JITter:SPECTrum:HORizontal:POSition?

The :JITter:SPECTrum:HORizontal:POSition? query returns the current center frequency in Hertz.

Return Format <value><NL>

<value> ::= X position (center), in NR3 format

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:SPECTrum:APPLY"](#) on page 598
 - [":JITter:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITter:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITter:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITter:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITter:SPECTrum:WINDow"](#) on page 604
 - [":JITter:STATe"](#) on page 606

:JITter:SPECTrum:HORizontal:RANGe

N (see [page 1708](#))

Command Syntax :JITter:SPECTrum:HORizontal:RANGe <value>
 <value> ::= X scaling (span), in NR3 format

The :JITter:SPECTrum:HORizontal:RANGe command sets the frequency span of the display (left graticule to right graticule) for the jitter measurement spectrum.

After the :JITter:SPECTrum:APPLY command, this command is an alias for :FUNCTION4[:FFT]:SPAN.

Query Syntax :JITter:SPECTrum:HORizontal:RANGe?

The :JITter:SPECTrum:HORizontal:RANGe? query returns the current frequency span in Hertz.

Return Format <value><NL>
 <value> ::= X scaling (span), in NR3 format

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:SPECTrum:APPLY"](#) on page 598
 - [":JITter:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITter:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITter:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITter:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITter:SPECTrum:WINDow"](#) on page 604
 - [":JITter:STATe"](#) on page 606

:JITter:SPECTrum:VERTical:OFFSet

N (see [page 1708](#))

Command Syntax :JITter:SPECTrum:VERTical:OFFSet <value>

<value> ::= Y offset, in NR3 format

The :JITter:SPECTrum:VERTical:OFFSet command sets the vertical value represented at center screen for the jitter measurement spectrum.

After the :JITter:SPECTrum:APPLY command, this command is an alias for :FUNCTION4:OFFSet.

Query Syntax :JITter:SPECTrum:VERTical:OFFSet?

The :JITter:SPECTrum:VERTical:OFFSet? query returns the current offset value for the jitter measurement spectrum.

Return Format <value><NL>

<value> ::= Y offset, in NR3 format

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:SPECTrum:APPLY"](#) on page 598
 - [":JITter:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITter:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITter:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITter:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITter:SPECTrum:WINDow"](#) on page 604
 - [":JITter:STATe"](#) on page 606

:JITTer:SPECTrum:VERTical:RANGe

N (see [page 1708](#))

Command Syntax :JITTer:SPECTrum:VERTical:RANGe <value>

<value> ::= Y scaling, in NR3 format

The :JITTer:SPECTrum:VERTical:RANGe command defines the full-scale vertical range for the jitter measurement spectrum.

After the :JITTer:SPECTrum:APPLy command, this command is an alias for :FUNCTION4:RANGe.

Query Syntax :JITTer:SPECTrum:VERTical:RANGe?

The :JITTer:SPECTrum:VERTical:RANGe? query returns the current full-scale vertical range for the jitter measurement spectrum.

Return Format <value><NL>

<value> ::= Y scaling, in NR3 format

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:SPECTrum:APPLy"](#) on page 598
 - [":JITTer:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITTer:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITTer:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITTer:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITTer:SPECTrum:WINDow"](#) on page 604
 - [":JITTer:STATe"](#) on page 606

:JITTer:SPECTrum:VERTical:TYPE

N (see [page 1708](#))

Command Syntax :JITTer:SPECTrum:VERTical:TYPE <type>

<type> ::= {DECibel | VRMS}

The :JITTer:SPECTrum:VERTical:TYPE command sets the vertical units for the jitter spectrum.

After the :JITTer:SPECTrum:APPLY command, this command is an alias for :FUNCTION4[:FFT]:VTYPE.

Query Syntax :JITTer:SPECTrum:VERTical:TYPE?

The :JITTer:SPECTrum:VERTical:TYPE? query returns the vertical units setting.

Return Format <type><NL>

<type> ::= {DEC | VRMS}

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:SPECTrum:APPLY"](#) on page 598
 - [":JITTer:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITTer:SPECTrum:HORizontal:RANGe"](#) on page 600
 - [":JITTer:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITTer:SPECTrum:VERTical:RANGe"](#) on page 602
 - [":JITTer:SPECTrum:WINDow"](#) on page 604
 - [":JITTer:STATe"](#) on page 606

:JITter:SPECTrum:WINDow

N (see [page 1708](#))

Command Syntax :JITter:SPECTrum:WINDow <window>

<window> ::= {RECTangular | HANNing | FLATtop | BHARris | BARTlett}

The :JITter:SPECTrum:WINDow command allows the selection of different windowing transforms or operations for the jitter spectrum FFT (Fast Fourier Transform).

The jitter spectrum 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).
- BARTlett – (triangular, with end points at zero) window is similar to the Hanning window in that it is good for making accurate frequency measurements, but its higher and wider secondary lobes make it not quite as good for resolving frequencies that are close together.

After the :JITter:SPECTrum:APPLY command, this command is an alias for :FUNCTION4[:FFT]:WINDow.

Query Syntax :JITter:SPECTrum:WINDow?

The :JITter:SPECTrum:WINDow? query returns the value of the window selected for the jitter spectrum function.

Return Format <window><NL>

<window> ::= {RECT | HANN | FLAT | BHAR | BART}

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:SPECTrum:APPLY"](#) on page 598
 - [":JITter:SPECTrum:HORizontal:POSition"](#) on page 599

- `":JITTer:SPECTrum:HORizontal:RANGe"` on page 600
- `":JITTer:SPECTrum:VERTical:OFFSet"` on page 601
- `":JITTer:SPECTrum:VERTical:RANGe"` on page 602
- `":JITTer:SPECTrum:VERTical:TYPE"` on page 603
- `":JITTer:STATe"` on page 606

:JITTer:STATe

 (see [page 1708](#))

Command Syntax :JITTer:STATe {{0 | OFF} | {1 | ON}}

The :JITTer:STATe command enables or disables the jitter analysis feature.

When jitter analysis is enabled, the **[Jitter]** front panel key is lit, and these oscilloscope settings are forced on:

- Precision measurements with a 1M analysis record length (see [":SYSTem:PRECision:LENGth"](#) on page 1443), which gives better accuracy.
- All edges in each acquisition are measured (instead of a single edge per acquisition, as is normally the case). For each acquisition, the measurement count increases by the number of edges in the acquisition. This applies to all measurements that look at edges, not just jitter analysis measurements.

When jitter analysis is disabled:

- The light behind the **[Jitter]** key is turned off.
- The "precision measurements" and 1M precision analysis record length settings are returned to the settings present when jitter analysis was enabled.
- The oscilloscope returns to measuring a specific edge in each acquisition (generally around the horizontal time reference) and ignoring all other edges in the acquisition, as is normally the case. For each acquisition, the measurement count increases by one.

Query Syntax :JITTer:STATe?

The :JITTer:STATe? query returns whether the jitter analysis feature is enabled.

Return Format <off_on><NL>
{0 | 1}

- See Also**
- [":JITTer:HISTogram:APPLY"](#) on page 595
 - [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:SPECTrum:APPLY"](#) on page 598
 - [":JITTer:SPECTrum:HORizontal:POSition"](#) on page 599
 - [":JITTer:SPECTrum:HORizontal:RANGE"](#) on page 600
 - [":JITTer:SPECTrum:VERTical:OFFSet"](#) on page 601
 - [":JITTer:SPECTrum:VERTical:RANGE"](#) on page 602
 - [":JITTer:SPECTrum:VERTical:TYPE"](#) on page 603
 - [":JITTer:SPECTrum:WINDow"](#) on page 604
 - [":JITTer:TREND:APPLY"](#) on page 608
 - [":JITTer:TREND:SMOoth:APPLY"](#) on page 609

- `":JITTer:TREND:SMOoth:POINts"` on page 610
- `":JITTer:TREND:VERTical:OFFSet"` on page 611
- `":JITTer:TREND:VERTical:RANGe"` on page 612

:JITTer:TREND:APPLY

N (see [page 1708](#))

Command Syntax :JITTer:TREND:APPLY

The :JITTer:TREND:APPLY command auto-configures and turns on the jitter measurement trend for the selected measurement.

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:STATe"](#) on page 606
 - [":JITTer:TREND:SMOoth:APPLY"](#) on page 609
 - [":JITTer:TREND:SMOoth:POINTs"](#) on page 610
 - [":JITTer:TREND:VERTical:OFFSet"](#) on page 611
 - [":JITTer:TREND:VERTical:RANGe"](#) on page 612

:JITTer:TREND:SMOoth:APPLy

N (see [page 1708](#))

Command Syntax :JITTer:TREND:SMOoth:APPLy

The :JITTer:TREND:SMOoth:APPLy command auto-configures and turns on smoothing for the jitter measurement trend.

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:STATe"](#) on page 606
 - [":JITTer:TREND:APPLy"](#) on page 608
 - [":JITTer:TREND:SMOoth:POINTs"](#) on page 610
 - [":JITTer:TREND:VERTical:OFFSet"](#) on page 611
 - [":JITTer:TREND:VERTical:RANGe"](#) on page 612

:JITter:TREnd:SMOoth:POINts

N (see [page 1708](#))

Command Syntax :JITter:TREnd:SMOoth:POINts <points>

<points> ::= odd integer in NR1 format

When jitter trend smoothing is turned on (by :JITter:TREnd:SMOoth:APPLy), the :JITter:TREnd: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.

After the :JITter:TREnd:SMOoth:APPLy command, this command is an alias for :FUNctioN3:SMOoth:POINts.

Query Syntax :JITter:TREnd:SMOoth:POINts?

The :JITter:TREnd:SMOoth:POINts? query returns the number of jitter trend smoothing points specified.

Return Format <points><NL>

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:STATe"](#) on page 606
 - [":JITter:TREnd:APPLy"](#) on page 608
 - [":JITter:TREnd:SMOoth:APPLy"](#) on page 609
 - [":JITter:TREnd:VERTical:OFFSet"](#) on page 611
 - [":JITter:TREnd:VERTical:RANGe"](#) on page 612

:JITter:TREND:VERTical:OFFSet

N (see [page 1708](#))

Command Syntax :JITter:TREND:VERTical:OFFSet <value>

<value> ::= Y offset, in NR3 format

The :JITter:TREND:VERTical:OFFSet command sets the vertical value represented at center screen for the jitter measurement trend.

After the :JITter:TREND:APPLY command, this command is an alias for :FUNCTION2:OFFSet.

Query Syntax :JITter:TREND:VERTical:OFFSet?

The :JITter:TREND:VERTical:OFFSet? query returns the current offset value for the jitter measurement trend.

Return Format <value><NL>

<value> ::= Y offset, in NR3 format

- See Also**
- [":JITter:MEASurement"](#) on page 596
 - [":JITter:STATe"](#) on page 606
 - [":JITter:TREND:APPLY"](#) on page 608
 - [":JITter:TREND:SMOoth:APPLY"](#) on page 609
 - [":JITter:TREND:SMOoth:POINTs"](#) on page 610
 - [":JITter:TREND:VERTical:RANGE"](#) on page 612

:JITTer:TREND:VERTical:RANGe

N (see [page 1708](#))

Command Syntax :JITTer:TREND:VERTical:RANGe <value>

<value> ::= Y scaling, in NR3 format

The :JITTer:TREND:VERTical:RANGe command defines the full-scale vertical range for the jitter measurement trend.

After the :JITTer:TREND:APPLY command, this command is an alias for :FUNCTION2:RANGe.

Query Syntax :JITTer:TREND:VERTical:RANGe?

The :JITTer:TREND:VERTical:RANGe? query returns the current full-scale vertical range for the jitter measurement trend.

Return Format <value><NL>

<value> ::= Y scaling, in NR3 format

- See Also**
- [":JITTer:MEASurement"](#) on page 596
 - [":JITTer:STATe"](#) on page 606
 - [":JITTer:TREND:APPLY"](#) on page 608
 - [":JITTer:TREND:SMOoth:APPLY"](#) on page 609
 - [":JITTer:TREND:SMOoth:POINTs"](#) on page 610
 - [":JITTer:TREND:VERTical:OFFSet"](#) on page 611

25 :LISTer Commands

Table 107 :LISTer Commands Summary

Command	Query	Options and Query Returns
n/a	:LISTer:DATA? (see page 614)	<binary_block> ::= comma-separated data with newlines at the end of each row
:LISTer:DISPlay { {OFF 0} {PBUS1 SBUS1 ON 1} {PBUS2 SBUS2 2} ALL} (see page 615)	:LISTer:DISPlay? (see page 615)	{OFF PBUS1 PBUS2 ALL}
:LISTer:REFeRence <time_ref> (see page 616)	:LISTer:REFeRence? (see page 616)	<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 1708](#))

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 613
 - [":LISTer:DISPlay"](#) on page 615
 - ["Definite-Length Block Response Data"](#) on page 228

:LISTer:DISPlay

N (see [page 1708](#))

Command Syntax :LISTer:DISPlay <value>

```
<value> ::= {{OFF | 0} | {PBUS1 | SBUS1 | ON | 1}
            | {PBUS2 | 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 displayed in different sections of the Protocol Listing.

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 | PBUS1 | PBUS2 | ALL}
```

- See Also**
- ["Introduction to :LISTer Commands"](#) on page 613
 - [":SBUS<n>:DISPlay"](#) on page 986
 - [":LISTer:DATA?"](#) on page 614

:LISTer:REFeRence

N (see [page 1708](#))

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 613
 - [":SBUS<n>:DISPlay"](#) on page 986
 - [":LISTer:DATA?"](#) on page 614
 - [":LISTer:DISPlay"](#) on page 615

26 :LTEST Commands

Table 108 :LTEST Commands Summary

Command	Query	Options and Query Returns
:LTEST:COPY (see page 619)	n/a	n/a
:LTEST:COPY:ALL (see page 620)	n/a	n/a
:LTEST:COPY:MARGin <percent> (see page 621)	:LTEST:COPY:MARGin? (see page 621)	<percent> ::= copy margin percent in NR1 format
:LTEST:ENABLE {{0 OFF} {1 ON}} (see page 622)	:LTEST:ENABLE? (see page 622)	<setting> ::= {0 1}
:LTEST:FAIL <condition> (see page 623)	:LTEST:FAIL? (see page 623)	<condition> ::= {INSide OUTSide}
:LTEST:LLIMit <lower_value> (see page 624)	:LTEST:LLIMit? (see page 624)	<lower_value> ::= a real number in NR3 format
:LTEST:MEASurement {MEAS<n>} (see page 625)	:LTEST:MEASurement? (see page 625)	<n> ::= # of installed measurement, from 1 to 10
n/a	:LTEST:RESults? [{MEAS<n>}] (see page 626)	<n> ::= # of installed measurement, from 1 to 10
:LTEST:RUMode:SOFailu re {{0 OFF} {1 ON}} (see page 627)	:LTEST:RUMode:SOFailu re? (see page 627)	<setting> ::= {0 1}

Table 108 :LTEST Commands Summary (continued)

Command	Query	Options and Query Returns
:LTEST:TEST {{0 OFF} {1 ON}} (see page 628)	:LTEST:TEST? (see page 628)	<setting> ::= {0 1}
:LTEST:ULIMit <upper_value> (see page 629)	:LTEST:ULIMit? (see page 629)	<upper_value> ::= a real number in NR3 format

Introduction to :LTEST Commands The LTEST subsystem is used for the Measurement Limit Test feature.

:LTEST:COPY

N (see [page 1708](#))

Command Syntax :LTEST:COPY

For the measurement selected by :LTEST:MEASurement, the :LTEST:COPY command sets upper and lower limits to the last measurement result plus or minus a percent margin (see :LTEST:COPY:MARGin).

- See Also**
- [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:COPY:ALL

N (see [page 1708](#))

Command Syntax

:LTEST:COPY:ALL

For all measurements whose limits are being tested, the :LTEST:COPY:ALL command sets upper and lower limits to the last measurement result plus or minus a percent margin (see :LTEST:COPY:MARGIN).

In essence, this command performs the :LTEST:COPY operation on all measurements whose limits are being tested.

- See Also
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:MARGIN"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:COPY:MARGIN

N (see [page 1708](#))

Command Syntax :LTEST:COPY:MARGIN <percent>

<percent> ::= copy margin percent in NR1 format

For the measurement selected by :LTEST:MEASUREMENT, the :LTEST:COPY:MARGIN command specifies the percent margin used when setting upper and lower limits by copying from measurement results.

Query Syntax :LTEST:COPY:MARGIN?

The :LTEST:COPY:MARGIN? query returns the percent margin setting for the measurement selected by :LTEST:MEASUREMENT.

Return Format <percent><NL>

<percent> ::= copy margin percent in NR1 format

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMIT"](#) on page 624
 - [":LTEST:MEASUREMENT"](#) on page 625
 - [":LTEST:RESULTS?"](#) on page 626
 - [":LTEST:RUMODE:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMIT"](#) on page 629

:LTEST:ENABLE

N (see [page 1708](#))

Command Syntax :LTEST:ENABLE {{0 | OFF} | {1 | ON}}

The :LTEST:ENABLE command turns the measurement limit test feature on or off.

Query Syntax :LTEST:ENABLE?

The :LTEST:ENABLE? query returns whether the measurement limit test feature is on or off.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGIN"](#) on page 621
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMIT"](#) on page 624
 - [":LTEST:MEASUREMENT"](#) on page 625
 - [":LTEST:RESULTS?"](#) on page 626
 - [":LTEST:RUMODE:SOFAILURE"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMIT"](#) on page 629

:LTEST:FAIL

N (see [page 1708](#))

Command Syntax :LTEST:FAIL <condition>
 <condition> ::= {INSide | OUTSide}

For the active measurement currently selected by the :LTEST:MEASurement command, the :LTEST:FAIL command sets the fail condition for the measurement.

When a measurement failure is detected by the limit test, the fail action conditions are executed, and there is the potential to generate an SRQ.

- INSide – causes the oscilloscope to fail a test when the measurement results are within the parameters set by the :LTEST:LIMit and :LTEST:ULIMit commands.
- OUTSide – causes the oscilloscope to fail a test when the measurement results exceed the parameters set by :LTEST:LLIMit and :LTEST:ULIMit commands.

Query Syntax :LTEST:FAIL?

The :LTEST:FAIL? query returns the currently set fail condition.

Return Format <condition><NL>
 <condition> ::= {INS | OUTS}

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:LLIMit

N (see [page 1708](#))

Command Syntax :LTEST:LLIMit <lower_value>
<lower_value> ::= a real number in NR3 format

For the active measurement currently selected by the :LTEST:MEASurement command, the :LTEST:LLIMit (Lower LIMit) command sets the lower test limit.

Query Syntax :LTEST:LLIMit?

The :LTEST:LLIMit? query returns the current lower limit setting.

Return Format <lower_value><NL>

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:MEASurement

N (see [page 1708](#))

Command Syntax :LTEST:MEASurement {MEAS<n>}

<n> ::= # of installed measurement, from 1 to 10

The :LTEST:MEASurement command selects the measurement source for the :LTEST:FAIL, :LTEST:LLIMit, :LTEST:ULIMit, and :LTEST:TEST commands. It selects one of the active measurements by its number, where MEAS1 is the first added measurement. When you add more than 10 measurements, measurement numbering starts again from MEAS1.

Query Syntax :LTEST:MEASurement?

The :LTEST:MEASurement? query returns the currently selected measurement source.

Return Format MEAS<n><NL>

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABle"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:RESults?

N (see [page 1708](#))

Query Syntax :LTEST:RESults? [{MEAS<n>}]

<n> ::= # of installed measurement, from 1 to 10

The :LTEST:RESults? query returns the measurement results for selected measurement.

When :LTEST:TEST is ON, the :LTEST:RESults? query returns the failed minimum value (Fail Min), the failed maximum value (Fail Max), and the total number of measurements made (# of Meas).

When :LTEST:TEST is OFF, the :LTEST:RESults? query returns nothing.

Return Format <fail_min>,<fail_max>,<num_meas><NL>

- <fail_min> – A real number representing the total number of measurements that have failed the minimum limit.
- <fail_max> – A real number representing the total number of measurements that have failed the maximum limit.
- <num_meas> – A real number representing the total number of measurements that have been made.

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABle"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMit"](#) on page 629

:LTEST:RUMode:SOFailure

N (see [page 1708](#))

Command Syntax :LTEST:RUMode:SOFailure {{0 | OFF} | {1 | ON}}

The :LTEST:RUMode:SOFailure command enables or disables the limit test "stop on failure" option.

When ON, the oscilloscope acquisition system stops once a limit failure is detected. If more than one measurement limit test is enabled, a failure of any of the measurements stops the oscilloscope from acquiring new waveforms.

Query Syntax :LTEST:RUMode:SOFailure?

The :LTEST:RUMode:SOFailure? query returns the "stop on failure" setting.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGIN"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMIT"](#) on page 624
 - [":LTEST:MEASUREMENT"](#) on page 625
 - [":LTEST:RESULTS?"](#) on page 626
 - [":LTEST:TEST"](#) on page 628
 - [":LTEST:ULIMIT"](#) on page 629

:LTEST:TEST

N (see [page 1708](#))

Command Syntax :LTEST:TEST {{0 | OFF} | {1 | ON}}

For the active measurement currently selected by the :LTEST:MEASUREMENT command, the :LTEST:TEST command enables or disables the limit test function on that measurement.

When any measurement has its limit test function enabled, the overall Limit Test feature is enabled.

The :LTEST:RESULTS? query returns nothing when :LTEST:TEST is OFF.

Query Syntax :LTEST:TEST?

The :LTEST:TEST? query returns the state of the TEST control for the active measurement currently selected by the :LTEST:MEASUREMENT command.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGIN"](#) on page 621
 - [":LTEST:ENABLE"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMIT"](#) on page 624
 - [":LTEST:MEASUREMENT"](#) on page 625
 - [":LTEST:RESULTS?"](#) on page 626
 - [":LTEST:RUMODE:SOFAILURE"](#) on page 627
 - [":LTEST:ULIMIT"](#) on page 629

:LTEST:ULIMit

N (see [page 1708](#))

Command Syntax :LTEST:ULIMit <upper_value>

<upper_value> ::= a real number in NR3 format

For the active measurement currently selected by the :LTEST:MEASurement command, the :LTEST:ULIMit (Upper LIMit) command sets the upper test limit.

Query Syntax :LTEST:ULIMit?

The :LTEST:ULIMit? query returns the current upper limit setting.

Return Format <upper_value><NL>

- See Also**
- [":LTEST:COPY"](#) on page 619
 - [":LTEST:COPY:ALL"](#) on page 620
 - [":LTEST:COPY:MARGin"](#) on page 621
 - [":LTEST:ENABle"](#) on page 622
 - [":LTEST:FAIL"](#) on page 623
 - [":LTEST:LLIMit"](#) on page 624
 - [":LTEST:MEASurement"](#) on page 625
 - [":LTEST:RESults?"](#) on page 626
 - [":LTEST:RUMode:SOFailure"](#) on page 627
 - [":LTEST:TEST"](#) on page 628

27 :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 633.

Table 109 :MARKer Commands Summary

Command	Query	Options and Query Returns
n/a	:MARKer:DYDX? (see page 634)	<return_value> ::= $\Delta Y / \Delta X$ value in NR3 format
:MARKer:MODE <mode> (see page 635)	:MARKer:MODE? (see page 635)	<mode> ::= {OFF MEASurement MANual WAVEform BINary HEX}
:MARKer:X1:DISPlay {{0 OFF} {1 ON}} (see page 636)	:MARKer:X1:DISPlay? (see page 636)	<setting> ::= {0 1}
:MARKer:X1Position <position>[suffix] (see page 637)	:MARKer:X1Position? (see page 637)	<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 page 638)	:MARKer:X1Y1source? (see page 638)	<source> ::= {CHANnel<n> FUNCTION<m> FFT MATH<m> WMEMory<r> HISTogram} <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 page 639)	:MARKer:X2:DISPlay? (see page 639)	<setting> ::= {0 1}

Table 109 :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
:MARKer:X2Position <position>[suffix] (see page 640)	:MARKer:X2Position? (see page 640)	<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 page 641)	:MARKer:X2Y2source? (see page 641)	<source> ::= {CHANnel<n> FUNCTION<m> FFT MATH<m> WMemory<r> HISTogram} <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 page 642)	<return_value> ::= X cursors delta value in NR3 format
:MARKer:XUNits <mode> (see page 643)	:MARKer:XUNits? (see page 643)	<units> ::= {SEConds HERTz DEGREes PERCent}
:MARKer:XUNits:USE (see page 644)	n/a	n/a
:MARKer:Y1:DISPlay {{0 OFF} {1 ON}} (see page 645)	:MARKer:Y1:DISPlay? (see page 645)	<setting> ::= {0 1}
:MARKer:Y1Position <position>[suffix] (see page 646)	:MARKer:Y1Position? (see page 646)	<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 page 647)	:MARKer:Y2:DISPlay? (see page 647)	<setting> ::= {0 1}
:MARKer:Y2Position <position>[suffix] (see page 648)	:MARKer:Y2Position? (see page 648)	<position> ::= Y2 cursor position value in NR3 format [suffix] ::= {V mV dB} <return_value> ::= Y2 cursor position value in NR3 format

Table 109 :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MARKer:YDELta? (see page 649)	<return_value> ::= Y cursors delta value in NR3 format
:MARKer:YUNits <mode> (see page 650)	:MARKer:YUNits? (see page 650)	<units> ::= {BASE PERCent}
:MARKer:YUNits:USE (see page 651)	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 1708](#))

Query Syntax :MARKer:DYDX?

The MARKer:DYDX? query returns the cursor $\Delta Y/\Delta X$ 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> ::= $\Delta Y/\Delta X$ value in NR3 format.

- See Also
- ["Introduction to :MARKer Commands"](#) on page 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:XUNits"](#) on page 643

:MARKer:MODE

N (see [page 1708](#))

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 633
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640
 - [":MARKer:Y1Position"](#) on page 646
 - [":MARKer:Y2Position"](#) on page 648

:MARKer:X1:DISPlay

N (see [page 1708](#))

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 636

:MARKer:X1Position

N (see [page 1708](#))

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 WAVEform (see [":MARKer:MODE"](#) on page 635).
- 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 WAVEform to put the cursors in the front-panel Normal mode.

Return Format <position><NL>
 <position> ::= X1 cursor position in NR3 format

- See Also**
- ["Introduction to :MARKer Commands"](#) on page 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X2Position"](#) on page 640
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:XUNits"](#) on page 643

:MARKer:X1Y1source

N (see [page 1708](#))

Command Syntax :MARKer:X1Y1source <source>

<source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | FFT | WMEMory<r>
| HISTogram}

<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 635):

- 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> | HIST | NONE}

- See Also**
- ["Introduction to :MARKer Commands"](#) on page 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733

:MARKer:X2:DISPlay

N (see [page 1708](#))

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 639

:MARKer:X2Position

N (see [page 1708](#))

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 WAVEform (see [":MARKer:MODE"](#) on page 635).
- 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 WAVEform 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 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:XUNits"](#) on page 643

:MARKer:X2Y2source

N (see [page 1708](#))

Command Syntax :MARKer:X2Y2source <source>

<source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | FFT | WMEMory<r> | HISTogram}

<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 635):

- 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> | HIST | NONE}

- See Also**
- ["Introduction to :MARKer Commands"](#) on page 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MEASure:SOURce"](#) on page 733

:MARKer:XDELta?

N (see [page 1708](#))

Query Syntax :MARKer:XDELta?

The MARKer:XDELta? query returns the value difference between the current X1 and X2 cursor positions.

$Xdelta = (\text{Value at X2 cursor}) - (\text{Value at X1 cursor})$

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 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:XUNits"](#) on page 643

:MARKer:XUNits

N (see [page 1708](#))

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 633
 - [":MARKer:XUNits:USE"](#) on page 644
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640

:MARKer:XUNits:USE

N (see [page 1708](#))

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 633
 - [":MARKer:XUNits"](#) on page 643
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733
 - [":MARKer:X1Position"](#) on page 637
 - [":MARKer:X2Position"](#) on page 640
 - [":MARKer:XDELta?"](#) on page 642

:MARKer:Y1:DISPlay

N (see [page 1708](#))

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>
<setting> ::= {0 | 1}

See Also • [":MARKer:Y1:DISPlay"](#) on page 645

:MARKer:Y1Position

N (see [page 1708](#))

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 635), 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 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:Y2Position"](#) on page 648
 - [":MARKer:YUNits"](#) on page 650

:MARKer:Y2:DISPlay

N (see [page 1708](#))

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 647

:MARKer:Y2Position

N (see [page 1708](#))

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 635), the :MARKer:Y2Position 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 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:Y1Position"](#) on page 646
 - [":MARKer:YUNits"](#) on page 650

:MARKer:YDELta?

N (see [page 1708](#))

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 633
 - [":MARKer:MODE"](#) on page 635
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MARKer:Y1Position"](#) on page 646
 - [":MARKer:Y2Position"](#) on page 648
 - [":MARKer:YUNits"](#) on page 650

:MARKer:YUNits

N (see [page 1708](#))

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 633
 - [":MARKer:YUNits:USE"](#) on page 651
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733
 - [":MARKer:Y1Position"](#) on page 646
 - [":MARKer:Y2Position"](#) on page 648
 - [":MARKer:YDELta?"](#) on page 649

:MARKer:YUNits:USE

N (see [page 1708](#))

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 633
 - [":MARKer:YUNits"](#) on page 650
 - [":MARKer:X1Y1source"](#) on page 638
 - [":MARKer:X2Y2source"](#) on page 641
 - [":MEASure:SOURce"](#) on page 733
 - [":MARKer:Y1Position"](#) on page 646
 - [":MARKer:Y2Position"](#) on page 648
 - [":MARKer:YDELta?"](#) on page 649

28 :MEASure Commands

Select automatic measurements to be made and control time markers. See ["Introduction to :MEASure Commands"](#) on page 671.

Table 110 :MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:AREa [<interval>] [, <source>] (see page 673)	:MEASure:AREa? [<interval>] [, <source>] (see page 673)	<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
:MEASure:BRATe [<source>] (see page 674)	:MEASure:BRATe? [<source>] (see page 674)	<source> ::= {<digital channels> CHANnel<n> FUNCTION<m> MATH<m> WMEMory<r>} <digital channels> ::= DIGital<d> <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 <return_value> ::= bit rate in Hz, NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:BWIDth [<source>] (see page 675)	:MEASure:BWIDth? [<source>] (see page 675)	<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 page 676)	n/a	n/a
:MEASure:COUNter [<source>] (see page 677)	:MEASure:COUNter? [<source>] (see page 677)	<source> ::= {CHANnel<n> DIGital<d> EXTErnal} <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:CTCNwidth [<source>] (see page 679)	:MEASure:CTCNwidth? [<source>] (see page 679)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= -Width to -Width jitter in NR3 format
:MEASure:CTCPwidth [<source>] (see page 681)	:MEASure:CTCPwidth? [<source>] (see page 681)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= +Width to +Width jitter in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:DElay [<source1>] [,<source2>] (see page 683)	:MEASure:DElay? [<source1>] [,<source2>] (see page 683)	<source1,2> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DElay:DEFine <source1_edge_slope>, <source1_edge_number> , <source1_edge_thresho ld>, <source2_edge_slope>, <source2_edge_number> , <source2_edge_thresho ld> (see page 685)	:MEASure:DElay:DEFine ? (see page 685)	<source1_edge_slope>, <source2_edge_slope> ::= {RISing FALLing} <source1_edge_number>, <source2_edge_number> ::= 0 to 1000 in NR1 format <source1_edge_threshold>, <source2_edge_threshold> ::= {LOWer MIDDle UPPer}
:MEASure:DUTYcycle [<source>] (see page 687)	:MEASure:DUTYcycle? [<source>] (see page 687)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FALLtime [<source>] (see page 688)	:MEASure:FALLtime? [<source>] (see page 688)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= time in seconds between the lower and upper thresholds in NR3 format
:MEASure:FFT:ACPR <chan_width>, <chan_spacing>, <chan>[,<source>] (see page 689)	:MEASure:FFT:ACPR? <chan_width>, <chan_spacing>, <chan>[,<source>] (see page 689)	<chan_width> ::= width of main range and sideband channels, Hz in NR3 format <chan_spacing> ::= spacing between main range and sideband channels, Hz in NR3 format <chan> ::= {CENTer HIGH<sb> LOW<sb>} <sb> ::= sideband 1 to 5 <source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= adjacent channel power ratio, dBV in NR3 format
:MEASure:FFT:CPOWer [<source>] (see page 690)	:MEASure:FFT:CPOWer? [<source>] (see page 690)	<source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= spectral channel power, dBV in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FFT:OBW <percentage>[,<source>] (see page 691)	:MEASure:FFT:OBW? <percentage>[,<source>] (see page 691)	<percentage> ::= percent of spectral power occupied bandwidth is measured for in NR3 format <source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= occupied bandwidth, Hz in NR3 format
:MEASure:FFT:THD [<source>] (see page 692)	:MEASure:FFT:THD? [<source>] (see page 692)	<source> ::= {FUNCTION<m> MATH<m> FFT} (source must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format <return_value> ::= total harmonic distortion ratio percent in NR3 format
:MEASure:FREQuency [<source>] (see page 693)	:MEASure:FREQuency? [<source>] (see page 693)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= frequency in Hertz in NR3 format
n/a	:MEASure:HISTogram:BWIDth? (see page 694)	<value> ::= bin width in NR3 format
n/a	:MEASure:HISTogram:HITS? (see page 695)	<value> ::= # of total hits in NR3 format
n/a	:MEASure:HISTogram:M1S? (see page 696)	<value> ::= % of hits within +/-1 std dev in NR3 format
n/a	:MEASure:HISTogram:M2S? (see page 697)	<value> ::= % of hits within +/-2 std devs in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:HISTogram:M3S? (see page 698)	<value> ::= % of hits within +/-3 std devs in NR3 format
n/a	:MEASure:HISTogram:MAXimum? (see page 699)	<value> ::= value of max bin in NR3 format
n/a	:MEASure:HISTogram:MEAN? (see page 700)	<value> ::= mean of histogram in NR3 format
n/a	:MEASure:HISTogram:MEDian? (see page 701)	<value> ::= median of histogram in NR3 format
n/a	:MEASure:HISTogram:MINimum? (see page 702)	<value> ::= value of min bin in NR3 format
n/a	:MEASure:HISTogram:MODE? (see page 703)	<value> ::= mode of histogram in NR3 format
n/a	:MEASure:HISTogram:PEAK? (see page 704)	<value> ::= # of hits in max bin in NR3 format
n/a	:MEASure:HISTogram:PEAK? (see page 705)	<value> ::= delta value between max bin and min bin in NR3 format
n/a	:MEASure:HISTogram:STDEViation? (see page 706)	<value> ::= standard deviation of histogram in NR3 format
:MEASure:HYSTeresis { {0 OFF} {1 ON} } (see page 707)	:MEASure:HYSTeresis? (see page 707)	{ 0 1 }
:MEASure:NCJitter <config> (see page 708)	:MEASure:NCJitter? <config> (see page 709)	<config> ::= <source>, <direction>, <num_periods> <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 <direction> ::= {RISing FALLing} <num_periods> ::= 1 to 10000 in NR1 format <return_value> ::= N-Period to N-Period jitter in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NDUTy [<source>] (see page 710)	:MEASure:NDUTy? [<source>] (see page 710)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= ratio of negative pulse width to period in NR3 format
:MEASure:NEDGes [<source>] (see page 711)	:MEASure:NEDGes? [<source>] (see page 711)	<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 falling edge count in NR3 format
:MEASure:NPERiod <config> (see page 712)	:MEASure:NPERiod? <config> (see page 712)	<config> ::= <source>, <slope>, <num_periods> <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 <slope> ::= {RISing FALLing} <num_periods> ::= 1 to 10000 in NR1 format <return_value> ::= N-Period jitter in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NPULses [<source>] (see page 713)	:MEASure:NPULses? [<source>] (see page 713)	<source> ::= {CHANnel<n> FUNction<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= the falling pulse count in NR3 format
:MEASure:NWIDth [<source>] (see page 714)	:MEASure:NWIDth? [<source>] (see page 714)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>] (see page 715)	:MEASure:OVERshoot? [<source>] (see page 715)	<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 percent of the overshoot of the selected waveform in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PEDGes [<source>] (see page 717)	:MEASure:PEDGes? [<source>] (see page 717)	<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 rising edge count in NR3 format
:MEASure:PERiod [<source>] (see page 718)	:MEASure:PERiod? [<source>] (see page 718)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source1> [,<source2>] (see page 719)	:MEASure:PHASe? [<source1> [,<source2>] (see page 719)	<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> ::= the phase angle value in degrees in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:PPULses [<source>] (see page 720)	:MEASure:PPULses? [<source>] (see page 720)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r> <digital channels>} <n> ::= 1 to (# analog channels) in NR1 format <m> ::= 1 to (# math functions) in NR1 format <r> ::= 1 to (# ref waveforms) in NR1 format <digital channels> ::= DIGital<d> <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= the rising pulse count in NR3 format
:MEASure:PREShoot [<source>] (see page 721)	:MEASure:PREShoot? [<source>] (see page 721)	<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 percent of preshoot of the selected waveform in NR3 format
:MEASure:PWIDth [<source>] (see page 722)	:MEASure:PWIDth? [<source>] (see page 722)	<source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= width of positive pulse in seconds in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:QUICK <source> (see page 723)	n/a	<source> ::= {CHANnel<n>}
n/a	:MEASure:RESults? <result_list> (see page 724)	<result_list> ::= comma-separated list of measurement results
:MEASure:RISetime [<source>] (see page 728)	:MEASure:RISetime? [<source>] (see page 728)	<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> ::= rise time in seconds in NR3 format
:MEASure:SDEviation [<source>] (see page 729)	:MEASure:SDEviation? [<source>] (see page 729)	<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 std deviation in NR3 format
:MEASure:SHOW {{0 OFF} {1 ON}} (see page 730)	:MEASure:SHOW? (see page 730)	{0 1}
:MEASure:SLEWrate [<source>[,<slope>]] (see page 731)	:MEASure:SLEWrate? [<source>[,<slope>]] (see page 731)	<source> ::= {CHANnel<n> FUNctIon<m> MATH<m> WMEMory<r>} <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

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:SOURce <source1> [,<source2>] (see page 733)	:MEASure:SOURce? (see page 733)	<source1,2> ::= {CHANnel<n> DIGital<d> FUNCTION<m> MATH<m> WMEMory<r> EXTernal} <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> ::= {<source> NONE}
:MEASure:STATistics <type> (see page 736)	:MEASure:STATistics? (see page 736)	<type> ::= {{ON 1} CURRent MEAN MINimum MAXimum STDDev COUNT} ON ::= all statistics returned
:MEASure:STATistics:D ISPlay {{0 OFF} {1 ON}} (see page 737)	:MEASure:STATistics:D ISPlay? (see page 737)	{0 1}
:MEASure:STATistics:I NCRement (see page 738)	n/a	n/a
:MEASure:STATistics:M COunt <setting> (see page 739)	:MEASure:STATistics:M COunt? (see page 739)	<setting> ::= {INFinite <count>} <count> ::= 2 to 2000 in NR1 format
:MEASure:STATistics:R ESet (see page 740)	n/a	n/a
:MEASure:STATistics:R SDeviation {{0 OFF} {1 ON}} (see page 741)	:MEASure:STATistics:R SDeviation? (see page 741)	{0 1}

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TEDGe [<slope>,<occurrence>,<source>] (see page 742)	:MEASure:TEDGe? [<slope>,<occurrence>,<source>] (see page 743)	<slope> ::= {RISing FALLing BOTH} <occurrence> ::= [+ -]<number> <number> ::= the edge number in NR1 format <source> ::= {<digital channels> CHANNEL<n> FUNCTION<m> MATH<m> WMemory<r>} <digital channels> ::= DIGital<d> <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 <return_value> ::= time in seconds of the specified transition
:MEASure:THResholds:ABSolute <upper>,<middle>,<lower> (see page 746)	:MEASure:THResholds:ABSolute? (see page 746)	<upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.
:MEASure:THResholds:METHod <threshold_mode> (see page 747)	:MEASure:THResholds:METHod? (see page 747)	<threshold_mode> ::= {PERCent ABSolute}
:MEASure:THResholds:PERCent <upper>,<middle>,<lower> (see page 748)	:MEASure:THResholds:PERCent? (see page 748)	<upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.
:MEASure:THResholds:STANdard (see page 749)	n/a	n/a

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TIEClock <config> (see page 750)	:MEASure:TIEClock? <config> (see page 750)	<config> ::= <source>, {SECond UNITinterval}, <direction> <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 <direction> ::= {RISing FALLing BOTH} <return_value> ::= Clock TIE jitter in NR3 format
:MEASure:TIEData <config> (see page 752)	:MEASure:TIEData? <config> (see page 752)	<config> ::= <source>, {SECond UNITinterval} <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> ::= Clock TIE jitter in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:TVALue? <value>, [<slope>]<occurrence> [,<source>] (see page 754)	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <source> ::= {CHANnel<n> DIGital<d> 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 <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 756)	:MEASure:VAMPlitude? [<source>] (see page 756)	<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 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAVerage [<interval>] [, <source>] (see page 757)	:MEASure:VAVerage? [<interval>] [, <source>] (see page 757)	<interval> ::= {CYCLe DISPlay} <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 <return_value> ::= calculated average voltage in NR3 format
:MEASure:VBASe [<source>] (see page 758)	:MEASure:VBASe? [<source>] (see page 758)	<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 <base_voltage> ::= voltage at the base of the selected waveform in NR3 format
:MEASure:VMAX [<source>] (see page 759)	:MEASure:VMAX? [<source>] (see page 759)	<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> ::= maximum voltage of the selected waveform in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VMIN [<source>] (see page 760)	:MEASure:VMIN? [<source>] (see page 760)	<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> ::= minimum voltage of the selected waveform in NR3 format
:MEASure:VPP [<source>] (see page 761)	:MEASure:VPP? [<source>] (see page 761)	<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> ::= voltage peak-to-peak of the selected waveform in NR3 format
:MEASure:VRATio [<interval>] [,<source 1>] [,<source2>] (see page 762)	:MEASure:VRATio? [<interval>] [,<source 1>] [,<source2>] (see page 762)	<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 <return_value> ::= the ratio value in dB in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VRMS [<interval>] [,<type>] [,<source>] (see page 763)	:MEASure:VRMS? [<interval>] [,<type>] [,<source>] (see page 763)	<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
:MEASure:VTOP [<source>] (see page 764)	:MEASure:VTOP? [<source>] (see page 764)	<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 page 765)	:MEASure:WINDow? (see page 765)	<type> ::= {MAIN ZOOM AUTO GATE}
:MEASure:XMAX [<source>] (see page 766)	:MEASure:XMAX? [<source>] (see page 766)	<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> ::= horizontal value of the maximum in NR3 format

Table 110 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:XMIN [<source>] (see page 767)	:MEASure:XMIN? [<source>] (see page 767)	<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> ::= horizontal value of the minimum in NR3 format
:MEASure:YATX <horiz_location>[,<source>] (see page 768)	:MEASure:YATX? <horiz_location>[,<source>] (see page 768)	<horiz_location> ::= displayed time from trigger in seconds in NR3 format <source> ::= {CHANnel<n> DIGital<d> 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 <d> ::= 0 to (# digital channels - 1) in NR1 format <return_value> ::= voltage at the specified time in NR3 format

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

Measurement Type	Portion of waveform that must be displayed
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:AREA

N (see [page 1708](#))

Command Syntax :MEASure:AREA [<interval>] [,<source>]

<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

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).

Query Syntax :MEASure:AREA? [<interval>] [,<source>]

The :MEASure:AREA? query measures and returns the area value.

Return Format <value><NL>

<value> ::= the area value in volt-seconds in NR3 format

See Also

- ["Introduction to :MEASure Commands"](#) on page 671
- [":MEASure:SOURce"](#) on page 733

:MEASure:BRATe

N (see [page 1708](#))

Command Syntax :MEASure:BRATe [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNCTion<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:FREQuency"](#) on page 693
 - [":MEASure:PERiod"](#) on page 718

:MEASure:BWIDth

N (see [page 1708](#))

Command Syntax :MEASure:BWIDth [<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: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:

$$\text{burst width} = (\text{last edge on screen} - \text{first edge on screen})$$

Return Format <value><NL>
 <value> ::= burst width in seconds in NR3 format

See Also

- ["Introduction to :MEASure Commands"](#) on page 671
- [":MEASure:SOURce"](#) on page 733

:MEASure:CLEar

 (see [page 1708](#))

Command Syntax :MEASure:CLEar

This command clears all selected measurements and markers from the screen.

See Also • ["Introduction to :MEASure Commands"](#) on page 671

:MEASure:COUNter

N (see [page 1708](#))

Command Syntax :MEASure:COUNter [<source>]
 <source> ::= {<digital channels> | CHANnel<n> | EXTernal}
 <digital channels> ::= DIGital<d>
 <n> ::= 1 to (# analog channels) in NR1 format
 <d> ::= 0 to (# digital channels - 1) in NR1 format

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 ≥ 0.1 s and ≤ 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 671
- [":MEASure:SOURce"](#) on page 733

- **":MEASure:FREQuency"** on page 693
- **":MEASure:CLEar"** on page 676

:MEASure:CTCNwidth

N (see [page 1708](#))

Command Syntax

NOTE

This measurement is available when the optional jitter analysis feature is licensed

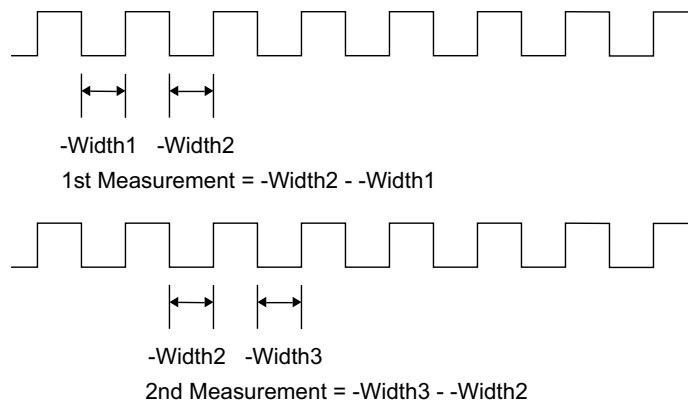
```
:MEASure:CTCNwidth [<source>]
```

```
<source> ::= CHANnel<n>
```

```
<n> ::= 1 to (# analog channels) in NR1 format
```

The :MEASure:CTCNwidth command installs a -Width to -Width (cycle-to-cycle negative width) jitter measurement on screen.

The -Width to -Width jitter measurement subtracts the first cycle's negative pulse width from the second cycle's negative pulse width for the first measurement result. Then, the second cycle's negative pulse width is subtracted from the third cycle's negative pulse width for the second measurement result, and so on, until all of the cycles of the waveform have been measured. The following figure shows the first two -Width to -Width measurements.



All of the measurement results are used to compute the statistics. The statistics are the values accumulated over all of the trigger cycles that have occurred.

The measurement is made on all the cycles of the waveform. Even in Zoom mode, all the cycles of the waveform in the Main window will be measured.

Query Syntax :MEASure:CTCNwidth? [<source>]

The :MEASure:CTCNwidth? query measures and returns the -Width to -Width jitter value.

Return Format <value><NL>

`<value> ::= -Width to -Width jitter in NR3 format`

- See Also
- [":MEASure:CTCPwidth"](#) on page 681
 - [Chapter 24](#), “:JITTer Commands,” starting on page 593

:MEASure:CTCPwidth

N (see [page 1708](#))

Command Syntax

NOTE

This measurement is available when the optional jitter analysis feature is licensed

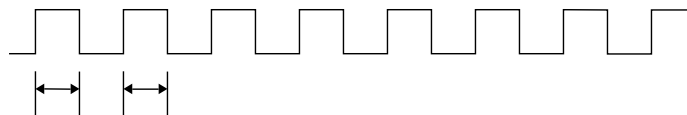
```
:MEASure:CTCPwidth [<source>]
```

```
<source> ::= CHANnel<n>
```

```
<n> ::= 1 to (# analog channels) in NR1 format
```

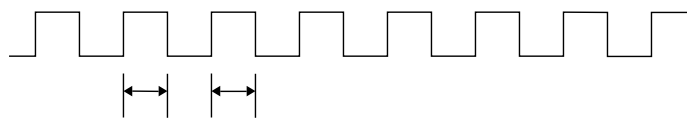
The :MEASure:CTCPwidth command installs a +Width to +Width (cycle-to-cycle positive width) jitter measurement on screen.

The +Width to +Width jitter measurement subtracts the first cycle's positive pulse width from the second cycle's positive pulse width for the first measurement result. Then, the second cycle's positive pulse width is subtracted from the third cycle's positive pulse width for the second measurement result, and so on, until all of the cycles of the waveform have been measured. The following figure shows the first two +Width to +Width measurements.



+Width1 +Width2

1st Measurement = +Width2 - +Width1



+Width2 +Width3

2nd Measurement = +Width3 - +Width2

All of the measurement results are used to compute the statistics. The statistics are the values accumulated over all of the trigger cycles that have occurred.

The measurement is made on all the cycles of the waveform. Even in Zoom mode, all the cycles of the waveform in the Main window will be measured.

Query Syntax :MEASure:CTCPwidth? [<source>]

The :MEASure:CTCPwidth? query measures and returns the +Width to +Width jitter value.

Return Format <value><NL>

<value> ::= +Width to +Width jitter in NR3 format

- See Also
- [":MEASure:CTCNwidth"](#) on page 679
 - [Chapter 24](#), ":JITTer Commands," starting on page 593

:MEASure:DElay

N (see [page 1708](#))

Command Syntax :MEASure:DElay [<edge_select_mode>][,][<source1>][,<source2>]

<edge_select_mode> ::= {MANual | AUTO}

<source1>, <source2> ::= {CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r> | <digital channels>}

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format

<digital channels> ::= DIGital<d>

<d> ::= 0 to (# digital channels - 1) 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 determined by the <edge_select_mode> and the :MEASure:DElay:DEfine command.

When the <edge_select_mode> is:

- AUTO – edges are automatically selected: the source1 edge closest to the timebase reference point is used, and the source2 edge closest to the source1 edge is used.
- MANual – edge numbers are determined by the :MEASure:DElay:DEfine settings.

Edge numbers greater than zero (0) are counted from the left side of the display for both sources.

Edge thresholds and slopes are always determined by the :MEASure:DElay:DEfine settings.

If the <edge_select_mode> is not included with the command, AUTO is the default.

Query Syntax :MEASure:DElay? [<edge_select_mode>][,][<source1>][,<source2>]

The :MEASure:DElay? query measures and returns the delay between source1 and source2. Delay measurement threshold, slope, and edge count parameters are determined by the <edge_select_mode> and the :MEASure:DElay:DEfine command.

The <edge_select_mode> definitions are the same as with the command syntax.

However, if the <edge_select_mode> is not included with the query, MANual is the default.

In the :MEASure:DEFine command, you can set upper, middle, and lower threshold values. The standard upper, middle, and lower measurement thresholds are 90%, 50%, and 10% values between Vbase and Vtop.

Return Format <value><NL>

<value> ::= floating-point number delay time in seconds in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:DElay:DEFine"](#) on page 685
 - [":MEASure:DElay:DEFine"](#) on page 685
 - [":MEASure:THResholds:ABSolute"](#) on page 746
 - [":MEASure:THResholds:METHod"](#) on page 747
 - [":MEASure:THResholds:PERCent"](#) on page 748
 - [":MEASure:THResholds:STANdard"](#) on page 749
 - [":MEASure:PHASe"](#) on page 719

:MEASure:DElay:DEFine

N (see [page 1708](#))

Command Syntax :MEASure:DElay:DEFine <source1_edge_slope>, <source1_edge_number>,
<source1_edge_threshold>, <source2_edge_slope>,
<source2_edge_number>, <source2_edge_threshold>

<source1_edge_slope>,
<source2_edge_slope> ::= {RISing | FALLing}

<source1_edge_number>,
<source2_edge_number> ::= 0 to 1000 in NR1 format

<source1_edge_threshold>,
<source2_edge_threshold> ::= {LOWer | MIDDle | UPPer}

The :MEASure:DElay:DEFine command defines slope directions and edge numbers for the delay measurement started or returned by the :MEASure:DElay command. The :MEASure:DElay:DEFine command also defines edge position parameters.

A <source1_edge_number> setting of zero (0) specifies that the edge closest to the timebase reference point automatically be selected. In this case, the <source2_edge_number> setting must also be zero (0), and the source2 edge closest to the selected source1 edge is used.

When edge numbers greater than zero (0) are specified, edges are counted from the left side of the display for both sources.

The selection of the source1 and source2 waveforms is made in the :MEASure:DElay or :MEASure:SOURce commands.

Query Syntax :MEASure:DElay:DEFine?

The :MEASure:DElay:DEFine? query returns the specified delay measurement parameter definitions.

Return Format <source1_edge_slope>, <source1_edge_number>,
<source1_edge_threshold>, <source2_edge_slope>,
<source2_edge_number>, <source2_edge_threshold><NL>

<source1_edge_slope>,
<source2_edge_slope> ::= {RIS | FALL}

<source1_edge_number>,
<source2_edge_number> ::= 0 to 1000 in NR1 format

<source1_edge_threshold>,
<source2_edge_threshold> ::= {LOW | MIDD | UPP}

- See Also**
- [":MEASure:DElay"](#) on page 683
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:THResholds:ABSolute"](#) on page 746

- **":MEASure:THResholds:METHod"** on page 747
- **":MEASure:THResholds:PERCent"** on page 748
- **":MEASure:THResholds:STANdard"** on page 749

:MEASure:DUTYcycle

C (see [page 1708](#))

Command Syntax :MEASure:DUTYcycle [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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:

$$+duty\ cycle = (+pulse\ width/period)*100$$

Return Format <value><NL>

<value> ::= ratio of positive pulse width to period in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:PERiod"](#) on page 718
 - [":MEASure:PWIDth"](#) on page 722
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:NDUTy"](#) on page 710

Example Code • ["Example Code"](#) on page 734

:MEASure:FALLtime

C (see [page 1708](#))

Command Syntax :MEASure:FALLtime [<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: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 timebase 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 671
 - [":MEASure:RISetime"](#) on page 728
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:SLEWrate"](#) on page 731

:MEASure:FFT:ACPR

N (see [page 1708](#))

Command Syntax :MEASure:FFT:ACPR <chan_width>,<chan_spacing>,<chan>[,<source>]
 <chan_width> ::= of main range and sideband channels, Hz in NR3 format
 <chan_spacing> ::= spacing between main range and sideband channels,
 Hz in NR3 format
 <chan> ::= {CENTer | HIGH<sb> | LOW<sb>}
 <sb> ::= sideband 1 to 5
 <source> ::= {FUNctIon<m> | MATH<m> | FFT} (must be an FFT waveform)
 <m> ::= 1 to (# math functions) in NR1 format

The :MEASure:FFT:ACPR command installs an FFT analysis Adjacent Channel Power Ratio (ACPR) measurement on screen.

Adjacent Channel Power Ratio (or channel leakage ratio) measures the ratio of the power in the main frequency range to the power contained in one or more sidebands.

The main range is specified by a channel width and a center frequency. The center frequency used in the measurement is the one defined for the FFT function.

Sidebands (with the same width as the main range) exist above and below the main range separated by the channel spacing width.

The sideband used for the measurement is selected with the <chan> parameter. You can select the first through fifth sidebands above or below the main range (HIGH1 through HIGH5 above and LOW1 through LOW5 below). The full sideband must be in the graticule (on screen) to be measured. Otherwise, the measurement results will be "Incomplete".

When this measurement is tracked with cursors, the cursors show the sideband being measured.

Query Syntax :MEASure:FFT:ACPR? <chan_width>,<chan_spacing>,<chan>[,<source>]

The :MEASure:FFT:ACPR? query returns the measured Adjacent Channel Power Ratio (ACPR) value.

Return Format <return_value><NL>
 <return_value> ::= adjacent channel power ratio, dBV in NR3 format

- See Also**
- [":MEASure:FFT:CPOWer"](#) on page 690
 - [":MEASure:FFT:OBW"](#) on page 691
 - [":MEASure:FFT:THD"](#) on page 692

:MEASure:FFT:CPOWer

N (see [page 1708](#))

Command Syntax :MEASure:FFT:CPOWer [<source>]
 <source> ::= {FUNCTION<m> | MATH<m> | FFT} (must be an FFT waveform)
 <m> ::= 1 to (# math functions) in NR1 format

The :MEASure:FFT:CPOWer command installs an FFT analysis Channel Power measurement on screen.

Channel Power measures the spectral power across a frequency range.

The center frequency used in the measurement is the one defined for the FFT function, and the FFT span specifies the frequency range.

When this measurement is tracked with cursors, the cursors are at the left and right edges of the frequency span.

Query Syntax :MEASure:FFT:CPOWer? [<source>]

The :MEASure:FFT:CPOWer? query returns the measured Channel Power value.

Return Format <return_value><NL>
 <return_value> ::= spectral channel power, dBV in NR3 format

- See Also**
- [":MEASure:FFT:ACPR"](#) on page 689
 - [":MEASure:FFT:OBW"](#) on page 691
 - [":MEASure:FFT:THD"](#) on page 692

:MEASure:FFT:OBW

N (see [page 1708](#))

Command Syntax :MEASure:FFT:OBW <percentage>[,<source>]

<percentage> ::= percent of spectral power occupied bandwidth is measured for (in NR3 format)

<source> ::= {FUNCTION<m> | MATH<m> | FFT} (must be an FFT waveform)

<m> ::= 1 to (# math functions) in NR1 format

The :MEASure:FFT:OBW command installs an FFT analysis Occupied Bandwidth measurement on screen.

Occupied Bandwidth measures the bandwidth (frequency range) containing some percent (usually 99%) of the total spectral power. While 99% is the industry norm, you can specify the percent you want to use in the measurement.

The center frequency used in the measurement is the one defined for the FFT function, and the FFT span represents the total spectral power.

When this measurement is tracked with cursors, the cursors show the measured bandwidth (frequency range).

Query Syntax :MEASure:FFT:OBW? <percentage>[,<source>]

The :MEASure:FFT:OBW? query returns the measured Occupied Bandwidth value.

Return Format <return_value><NL>

<return_value> ::= occupied bandwidth, Hz in NR3 format

- See Also**
- [":MEASure:FFT:ACPR"](#) on page 689
 - [":MEASure:FFT:CPOWer"](#) on page 690
 - [":MEASure:FFT:THD"](#) on page 692

:MEASure:FFT:THD

N (see [page 1708](#))

Command Syntax :MEASure:FFT:THD <tracking>[,<fundamental_freq>][,<source>]

<tracking> ::= {AUTO | MANual}

<fundamental_freq> ::= in NR3 format, required if <tracking> is MANual

<source> ::= {FUNCTION<m> | MATH<m> | FFT} (must be an FFT waveform)

<m> ::= 1 to (# math functions) in NR1 format

The :MEASure:FFT:THD command installs an FFT analysis Total Harmonic Distortion measurement on screen.

Total Harmonic Distortion (THD) is the ratio of power in the fundamental frequency to the power contained in the rest of the harmonics and noise. THD is a measure of signal purity.

Total Harmonic Distortion (THD) measures the power contained in the bands surrounding each harmonic and compares it to the power in the band surrounding the fundamental frequency. The width of the bands measured is the same for the fundamental frequency and each harmonic. That width is 1/2 of the fundamental frequency.

You can either enter the fundamental frequency as a measurement parameter and have the fundamental frequency and harmonics be tracked manually, or you can allow the fundamental frequency and harmonics to be tracked automatically, where the highest peak is assumed to be the fundamental frequency.

When this measurement is tracked with cursors, the cursors show the band surrounding the fundamental frequency that is being measured (at $\pm 1/4$ of the fundamental frequency).

Query Syntax :MEASure:FFT:THD? <tracking>[,<fundamental_freq>][,<source>]

The :MEASure:FFT:THD? query returns the measured Total Harmonic Distortion value.

Return Format <return_value><NL>

<return_value> ::= total harmonic distortion ratio percent in NR3 format

- See Also**
- [":MEASure:FFT:ACPR"](#) on page 689
 - [":MEASure:FFT:CPOWer"](#) on page 690
 - [":MEASure:FFT:OBW"](#) on page 691
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713
 - ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

:MEASure:FREQuency

C (see [page 1708](#))

Command Syntax :MEASure:FREQuency [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
- [":MEASure:SOURce"](#) on page 733
- [":MEASure:PERiod"](#) on page 718

Example Code

- ["Example Code"](#) on page 734

:MEASure:HISTogram:BWIDth?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:BWIDth?

The :MEASure:HISTogram:BWIDth? query returns the width of each bin (bucket) in the histogram.

Return Format <value><NL>

<value> ::= bin width in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:HITS?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:HITS?

The :MEASure:HISTogram:HITS? query returns the sum of all hits in all bins (buckets) in the histogram.

Return Format <value><NL>

<value> ::= # of total hits in NR1 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:M1S?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:M1S?

The :MEASure:HISTogram:M1S? query returns the percentage of histogram hits that lie within ± 1 Std Dev of the mean.

Return Format <value><NL>

<value> ::= % of hits within +/-1 std dev in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:M2S?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:M2S?

The :MEASure:HISTogram:M2S? query returns the percentage of histogram hits that lie within ± 2 Std Dev of the mean.

Return Format <value><NL>

<value> ::= % of hits within +/-2 std devs in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:M3S?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:M3S?

The :MEASure:HISTogram:M3S? query returns the percentage of histogram hits that lie within ± 3 Std Dev of the mean.

Return Format <value><NL>

<value> ::= % of hits within +/-3 std devs in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:MAXimum?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:MAXimum?

The :MEASure:HISTogram:MAXimum? query returns the value corresponding to the maximum bin that has any hits.

Return Format <value><NL>

<value> ::= value of max bin in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:MEAN?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:MEAN?

The :MEASure:HISTogram:MEAN? query returns the mean value of the histogram.

Return Format <value><NL>

<value> ::= mean of histogram in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:MEDian?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:MEDian?

The :MEASure:HISTogram:MEDian? query returns the median value of the histogram.

Return Format <value><NL>

<value> ::= median of histogram in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:MINimum?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:MINimum?

The :MEASure:HISTogram:MINimum? query returns the value corresponding to the minimum bin that has any hits.

Return Format <value><NL>

<value> ::= value of min bin in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:MODE?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:MODE?

The :MEASure:HISTogram:MODE? query returns the mode value of the histogram.

Return Format <value><NL>

<value> ::= mode of histogram in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:PEAK?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:PEAK?

The :MEASure:HISTogram:PEAK? query returns the maximum number of hits in any single bin.

Return Format <value><NL>

<value> ::= # of hits in max bin in NR1 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PPEak?"](#) on page 705
 - [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:PPEak?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:PPEak?

The :MEASure:HISTogram:PPEak? query returns the delta between the Max and Min values.

Return Format <value><NL>

<value> ::= delta value between max bin and min bin in NR3 format

See Also • [Chapter 23](#), “:HISTogram Commands,” starting on page 577

- [":MEASure:HISTogram:BWIDth?"](#) on page 694
- [":MEASure:HISTogram:HITS?"](#) on page 695
- [":MEASure:HISTogram:M1S?"](#) on page 696
- [":MEASure:HISTogram:M2S?"](#) on page 697
- [":MEASure:HISTogram:M3S?"](#) on page 698
- [":MEASure:HISTogram:MAXimum?"](#) on page 699
- [":MEASure:HISTogram:MEAN?"](#) on page 700
- [":MEASure:HISTogram:MEDian?"](#) on page 701
- [":MEASure:HISTogram:MINimum?"](#) on page 702
- [":MEASure:HISTogram:MODE?"](#) on page 703
- [":MEASure:HISTogram:PEAK?"](#) on page 704
- [":MEASure:HISTogram:SDEViation?"](#) on page 706

:MEASure:HISTogram:SDEViation?

N (see [page 1708](#))

Query Syntax :MEASure:HISTogram:SDEViation?

The :MEASure:HISTogram:SDEViation? query returns the standard deviation of the histogram values.

Return Format <value><NL>

<value> ::= standard deviation of histogram in NR3 format

- See Also**
- [Chapter 23](#), “:HISTogram Commands,” starting on page 577
 - [":MEASure:HISTogram:BWIDth?"](#) on page 694
 - [":MEASure:HISTogram:HITS?"](#) on page 695
 - [":MEASure:HISTogram:M1S?"](#) on page 696
 - [":MEASure:HISTogram:M2S?"](#) on page 697
 - [":MEASure:HISTogram:M3S?"](#) on page 698
 - [":MEASure:HISTogram:MAXimum?"](#) on page 699
 - [":MEASure:HISTogram:MEAN?"](#) on page 700
 - [":MEASure:HISTogram:MEDian?"](#) on page 701
 - [":MEASure:HISTogram:MINimum?"](#) on page 702
 - [":MEASure:HISTogram:MODE?"](#) on page 703
 - [":MEASure:HISTogram:PEAK?"](#) on page 704
 - [":MEASure:HISTogram:PPEak?"](#) on page 705

:MEASure:HYSTeresis

N (see [page 1708](#))

Command Syntax :MEASure:HYSTeresis {{0 | OFF} | {1 | ON}}

The :MEASure:HYSTeresis command disables or enables the minimum signal amplitude limit for making measurements.

When the oscilloscope makes automatic parametric measurements on captured waveform data, it builds a vertical histogram, recording how many times each Q-level (quantization level, an internal voltage representation) is present in the data record. From the histogram, the absolute maximum and minimum voltage levels, as well as relative maximum (top) and minimum (base) voltage levels, are determined. Using the top and base levels, threshold levels are calculated. The waveform data record is again analyzed using the thresholds to determine signal edges.

To prevent the oscilloscope from making incorrect measurements on system noise, there is a minimum signal amplitude (in Q-levels, as a fraction of the vertical division) below which the measurement process does not take place, and "Low signal" is reported in the measurement results.

However, with the ability to limit channel bandwidth, the amount of system noise can vary, and with the oscilloscope's low noise floor, it is possible to get a very small amplitude signal on screen that is stable, and therefore, theoretically measureable, even though it falls below the minimum signal amplitude limit. In these cases, you can turn off the measurement hysteresis limit to remove the "Low signal" result and make measurements.

The default measurement hysteresis setting (enabled) delivers optimal measurement performance for almost all signals.

Query Syntax :MEASure:HYSTeresis?

The :MEASure:HYSTeresis? query returns the state of the measurement hysteresis setting.

Return Format {0 | 1}<NL>

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:RESults?"](#) on page 724

:MEASure:NCJitter

N (see [page 1708](#))

Command Syntax

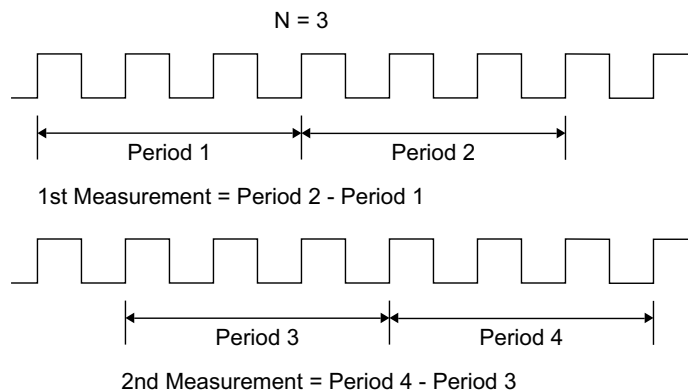
NOTE

This measurement is available when the optional jitter analysis feature is licensed

```
:MEASure:NCJitter <config>
<config> ::= <source>, <direction>, <num_periods>
<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
<direction> ::= {RISing | FALLing}
<num_periods> ::= 1 to 10000 in NR1 format
```

The :MEASure:NCJitter command install a period-period (N-cycle) jitter measurement on screen.

The Period-Period jitter measurement takes a group of cycles and measures the period from the first edge of the first cycle of the group to the final edge of the last cycle of the group. The N-period of the first group is measured and subtracted from the N-period of the second group. After shifting ahead one cycle the next measurement is made. The following figure shows how an N-cycle jitter measurement is made for N=3.



Once all of the groups of cycles of the waveform are measured, the statistics are calculated. The statistics are the values accumulated over all of the trigger cycles that have occurred.

To perform a cycle-cycle jitter measurement, use Period-Period and set N=1.

The measurement is made on all the cycles of the waveform. Even in Zoom mode, all the cycles of the waveform in the Main window will be measured.

Query Syntax :MEASure:NCJitter? <config>

The :MEASure:NCJitter? query measures and returns the N-Period to N-Period jitter value.

Return Format

<value><NL>

<value> ::= N-Period to N-Period jitter in NR3 format

- See Also**
- **":MEASure:NPERiod"** on page 712
 - **Chapter 24**, "JITTer Commands," starting on page 593

:MEASure:NDUTy

N (see [page 1708](#))

Command Syntax :MEASure:NDUTy [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
 - [":MEASure:PERiod"](#) on page 718
 - [":MEASure:NWIDth"](#) on page 714
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:DUTYcycle"](#) on page 687

:MEASure:NEDGes

N (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:NPERiod

N (see [page 1708](#))

Command Syntax

NOTE

This measurement is available when the optional jitter analysis feature is licensed

```

:MEASure:NPERiod <config>

<config> ::= <source>, <slope>, <num_periods>

<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

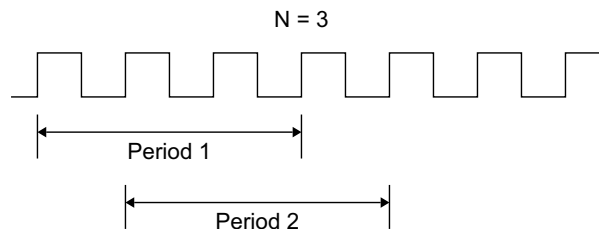
<slope> ::= {RISing | FALLing}

<num_periods> ::= 1 to 10000 in NR1 format

```

The :MEASure:NPERiod command installs an N-period jitter measurement on screen.

The N-Period jitter measurement measures the time span of N consecutive periods. Then, it shifts ahead one period and measures the time span of the next N consecutive periods, and so on.



Query Syntax :MEASure:NPERiod? <config>

The :MEASure:NPERiod? query measures and returns the N-Period jitter value.

Return Format

```

<value><NL>

<value> ::= N-Period jitter in NR3 format

```

- See Also**
- [":MEASure:NCJitter"](#) on page 708
 - [Chapter 24](#), "JITTER Commands," starting on page 593

:MEASure:NPULses

N (see [page 1708](#))

Command Syntax :MEASure:NPULses [<source>]

<source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | WMEMory<r>
| <digital channels>}

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format

<digital channels> ::= DIGital<d>

<d> ::= 0 to (# digital channels - 1) 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 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:NWIDth

C (see [page 1708](#))

Command Syntax :MEASure:NWIDth [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:PWIDth"](#) on page 722
 - [":MEASure:PERiod"](#) on page 718

:MEASure:OVERshoot

C (see [page 1708](#))

Command Syntax :MEASure:OVERshoot [<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: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:

$$\text{overshoot} = ((V_{\text{max}} - V_{\text{top}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

For a falling edge:

$$\text{overshoot} = ((V_{\text{base}} - V_{\text{min}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

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 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 671
 - [":MEASure:PREShoot"](#) on page 721
 - [":MEASure:SOURce"](#) on page 733

- **":MEASure:VMAX"** on page 759
- **":MEASure:VTOP"** on page 764
- **":MEASure:VBASe"** on page 758
- **":MEASure:VMIN"** on page 760

:MEASure:PEDGes

N (see [page 1708](#))

Command Syntax :MEASure:PEDGes [<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: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 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:PERiod

C (see [page 1708](#))

Command Syntax :MEASure:PERiod [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m> | WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:NWIDth"](#) on page 714
 - [":MEASure:PWIDth"](#) on page 722
 - [":MEASure:FREQuency"](#) on page 693

Example Code • ["Example Code"](#) on page 734

:MEASure:PHASe

N (see [page 1708](#))

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 671
 - [":MEASure:DElay"](#) on page 683
 - [":MEASure:PERiod"](#) on page 718
 - [":MEASure:SOURce"](#) on page 733

:MEASure:PPULses

N (see [page 1708](#))

Command Syntax :MEASure:PPULses [<source>]

<source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | WMEMory<r>
| <digital channels>}

<n> ::= 1 to (# analog channels) in NR1 format

<m> ::= 1 to (# math functions) in NR1 format

<r> ::= 1 to (# ref waveforms) in NR1 format

<digital channels> ::= DIGital<d>

<d> ::= 0 to (# digital channels - 1) 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 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:PREShoot

C (see [page 1708](#))

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:

$$\text{preshoot} = ((V_{\min} - V_{\text{base}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

For a falling edge:

$$\text{preshoot} = ((V_{\max} - V_{\text{top}}) / (V_{\text{top}} - V_{\text{base}})) \times 100$$

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VMIN"](#) on page 760
 - [":MEASure:VMAX"](#) on page 759
 - [":MEASure:VTOP"](#) on page 764
 - [":MEASure:VBASe"](#) on page 758

:MEASure:PWIDth

C (see [page 1708](#))

Command Syntax :MEASure:PWIDth [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| WMEMory<r>}

<digital channels> ::= DIGital<d>

<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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:NWIDth"](#) on page 714
 - [":MEASure:PERiod"](#) on page 718

:MEASure:QUICK

N (see [page 1708](#))

Command Syntax :MEASure:QUICK <source>
 <source> ::= {CHANnel<n>}

This command installs 10 common measurements on an analog input channel source.

See Also • ["Introduction to :MEASure Commands"](#) on page 671

:MEASure:RESults?

N (see [page 1708](#))

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 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 671
 - [":MEASure:STATistics"](#) on page 736
 - ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

Example Code

```
' This program shows the InfiniiVision oscilloscope measurement
' results command.
' -----
```

Option Explicit

```
Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String
```



```

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Lo
ng)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 10000 ' Set I/O communication timeout.

    ' 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 CHANnel2" ' Input source.

    Dim MeasurementArray(3) As String
    MeasurementArray(0) = "FREQuency"
    MeasurementArray(1) = "DUTYcycle"
    MeasurementArray(2) = "VAMPlitude"
    MeasurementArray(3) = "VPP"
    Dim Measurement As Variant

    For Each Measurement In MeasurementArray
        myScope.WriteString ":MEASure:" + Measurement
        myScope.WriteString ":MEASure:" + Measurement + "?"
        strQueryResult = myScope.ReadString ' Read measurement value.
        Debug.Print Measurement + ": " + strQueryResult
    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

Next

Exit Sub

```

```
VisaComError:  
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description  
  
End Sub
```

:MEASure:RISetime

C (see [page 1708](#))

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 timebase 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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:FALLtime"](#) on page 688
 - [":MEASure:SLEWrate"](#) on page 731

:MEASure:SDEViation

N (see [page 1708](#))

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 671
 - [":MEASure:VRMS"](#) on page 763
 - [":MEASure:SOURce"](#) on page 733

:MEASure:SHOW

N (see [page 1708](#))

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 671
 - [":MARKer:MODE"](#) on page 635

:MEASure:SLEWrate

N (see [page 1708](#))

Command Syntax :MEASure:SLEWrate [<source>[,<slope>]]

<source> ::= {CHANnel<n> | FUNction<m> | MATH<m> | WMEMory<r>}

<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

<slope> ::= {RISing | FALLing | EITHer}

The :MEASure:SLEWrate command installs a slew rate measurement on the oscilloscope's front-panel screen.

The slew rate of an edge is the change in vertical value between the lower and upper thresholds of a waveform source (typically volts) divided by the change in time between where the waveform threshold crossings take place.

The lower and upper thresholds for the measurement are specified by the measurement threshold settings for the source waveform (see :MEASure:DEFine). If the <source> parameter is not specified, the current :MEASure:SOURce setting is used.

The on-screen edge closest to the timebase reference is measured.

You can specify that a RISing edge be measured, a FALLing edge be measured, or either a rising or a falling edge be measured (EITHer). If the <slope> is not specified, a rising edge is measured.

NOTE

This command is not available if the source is a FFT (Fast Fourier Transform) waveform.

Query Syntax :MEASure:SLEWrate? [<source>[,<slope>]]

The :MEASure:SLEWrate? query returns the measured slew rate of the specified edge closest to the timebase reference.

Return Format <value><NL>

<value> ::= slew rate in vertical units/second in NR3 format

- See Also**
- [":MEASure:DElay:DEFine"](#) on page 685
 - [":MEASure:THResholds:ABSolute"](#) on page 746
 - [":MEASure:THResholds:METHod"](#) on page 747
 - [":MEASure:THResholds:PERCent"](#) on page 748
 - [":MEASure:THResholds:STANdard"](#) on page 749

- [":MEASure:SOURce"](#) on page 733
- [":MEASure:RISetime"](#) on page 728
- [":MEASure:FALLtime"](#) on page 688
- ["Introduction to :MEASure Commands"](#) on page 671

:MEASure:SOURce

C (see [page 1708](#))

Command Syntax :MEASure:SOURce <source1>[,<source2>]

<source1>,<source2> ::= {<digital channels> | CHANnel<n> | FUNcTion<m>
| MATH<m> | WMEMory<r>}

<digital channels> ::= DIGital<d>

<n> ::= 1 to (# of analog channels) in NR1 format

<m> ::= 1 to (# math functions) 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<m>, or MATH<m> will also set :MARKer:X1Y1source to source1 and :MARKer:X2Y2source to source2.

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 applies only to :MEASure:DElay and :MEASure:PHASe measurements.

NOTE

MATH<m> is an alias for FUNcTion<m>. The query will return FUNC<m> if the source is FUNcTion<m> or MATH<m>.

NOTE

FUNcTion or MATH (without the "<m>" math function number) is an alias for FUNcTion2 or MATH2. The query will return FUNC2 if the source is FUNcTion or MATH.

Return Format <source1>,<source2><NL>

<source1>,<source2> ::= {<digital channels> | CHAN<n> | FUNC<m>
| WMEM<r> | NONE}

- See Also:**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MARKer:MODE"](#) on page 635

- **":MARKer:X1Y1source"** on page 638
- **":MARKer:X2Y2source"** on page 641
- **":MEASure:DElay"** on page 683
- **":MEASure:PHASe"** on page 719

Example Code

```
' This program shows InfiniiVision oscilloscope measure commands.
' -----

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Lo
ng)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 10000 ' Set I/O communication timeout.

    ' 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"

    ' 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"

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

See complete example programs at: [Chapter 49](#), "Programming Examples," starting on page 1717

:MEASure:STATistics

N (see [page 1708](#))

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 671
 - [":MEASure:RESults?"](#) on page 724
 - [":MEASure:STATistics:DISPlay"](#) on page 737
 - [":MEASure:STATistics:RESet"](#) on page 740
 - [":MEASure:STATistics:INCRement"](#) on page 738

Example Code • ["Example Code"](#) on page 724

:MEASure:STATistics:DISPlay

N (see [page 1708](#))

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 671
 - [":MEASure:RESults?"](#) on page 724
 - [":MEASure:STATistics"](#) on page 736
 - [":MEASure:STATistics:MCOunt"](#) on page 739
 - [":MEASure:STATistics:RESet"](#) on page 740
 - [":MEASure:STATistics:INCRement"](#) on page 738
 - [":MEASure:STATistics:RSDeviation"](#) on page 741

:MEASure:STATistics:INCRement

N (see [page 1708](#))

Command Syntax :MEASure:STATistics:INCRement

This command updates the statistics once (incrementing the count by one) using the current measurement values. 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 allowed only when the oscilloscope is stopped and measurements are on.

The command is allowed in segmented acquisition mode.

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:STATistics"](#) on page 736
 - [":MEASure:STATistics:DISPlay"](#) on page 737
 - [":MEASure:STATistics:RESet"](#) on page 740
 - [":MEASure:RESults?"](#) on page 724

:MEASure:STATistics:MCOunt

N (see [page 1708](#))

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 671
 - [":MEASure:RESults?"](#) on page 724
 - [":MEASure:STATistics"](#) on page 736
 - [":MEASure:STATistics:DISPlay"](#) on page 737
 - [":MEASure:STATistics:RSDeviation"](#) on page 741
 - [":MEASure:STATistics:RESet"](#) on page 740
 - [":MEASure:STATistics:INCRement"](#) on page 738

:MEASure:STATistics:RESet

N (see [page 1708](#))

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 671
 - [":MEASure:STATistics"](#) on page 736
 - [":MEASure:STATistics:DISPlay"](#) on page 737
 - [":MEASure:RESults?"](#) on page 724
 - [":MEASure:STATistics:INCRement"](#) on page 738

- Example Code**
- ["Example Code"](#) on page 724

:MEASure:STATistics:RSDeviation

N (see [page 1708](#))

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 671
 - [":MEASure:RESults?"](#) on page 724
 - [":MEASure:STATistics"](#) on page 736
 - [":MEASure:STATistics:DISPlay"](#) on page 737
 - [":MEASure:STATistics:MCOunt"](#) on page 739
 - [":MEASure:STATistics:RESet"](#) on page 740
 - [":MEASure:STATistics:INCRement"](#) on page 738

:MEASure:TEDGE

N (see [page 1708](#))

Command Syntax :MEASure:TEDGE [<slope>,<occurrence>[,<source>]

<slope> ::= {RISing | FALLing | BOTH}

<occurrence> ::= [+ | -]<number>

<number> ::= the edge number in NR1 format

<source> ::= {<digital channels> | CHANnel<n> | FUNCTION<m> | MATH<m>
| WMemory<r>}

<digital channels> ::= DIGital<d>

<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

The :MEASure:TEDGE command installs a screen measurement whose result is the time of the specified edge transition relative to the trigger edge (time=zero).

The threshold voltage used for this measurement is at the 50% point with a small amount of hysteresis added. You cannot change the threshold voltage used for this measurement with the :MEASure:DEFine command.

The optional <slope> parameter specifies whether to look for RISing edges or FALLing edges. If not otherwise specified, rising edges are assumed.

The <occurrence> parameter specifies the Nth edge to identify.

An <occurrence> value of zero (0) means the edge closest to the timebase reference; it is equivalent to the **Auto** selection in the graphical user interface. The BOTH (both edges) <slope> option can be used only when the <occurrence> count is zero (0).

Any other <occurrence> value means the Nth edge from the left side of the display.

The <occurrence> parameter can optionally have a plus sign (+) or a minus sign (-) to specify a rising or falling slope. Plus or minus signs cannot be used when the <slope> parameter is used.

If the optional <source> parameter is not specified, the current :MEASure:SOURce setting is used.

NOTE

This command is not available if the source is a FFT (Fast Fourier Transform) waveform.

Query Syntax :MEASure:TEDGe? [<slope>,<occurrence>[,<source>]

When the :MEASure:TEDGe query is sent, the displayed signal is searched for the specified transition.

The time interval between the trigger event and the edge occurrence is returned.

If the specified crossing cannot be found, the oscilloscope returns +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.

Return Format <value><NL>

<value> ::= time in seconds of the specified transition in NR3 format

:MEASure:TEDGe Code You can use multiple :MEASure:TEDGe? measurements to form delay and phase measurements:

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 example:

```
' This program shows the InfiniiVision oscilloscope :MEASure:TEDGe
' command.
' -----
```

Option Explicit

```
Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 10000 ' Set I/O communication timeout.

    ' 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"

' 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)

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

```

See complete example programs at: [Chapter 49](#), “Programming Examples,” starting on page 1717

See Also • ["Introduction to :MEASure Commands"](#) on page 671

- **":MEASure:TVALue?"** on page 754
- **":MEASure:YATX"** on page 768
- **"Commands Not Supported in Multiple Program Message Units"** on page 1713
- **"Queries Not Supported in Multiple Program Message Units"** on page 1713

:MEASure:THResholds:ABSolute

N (see [page 1708](#))

Command Syntax :MEASure:THResholds:ABSolute <upper>,<middle>,<lower>[,<source>]
 <upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.
 <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:THResholds:ABSolute command sets the upper, middle, and lower measurement thresholds to absolute values.

ABSolute thresholds are dependent on channel scaling (:CHANnel<n>:RANGe or :CHANnel<n>:SCALE), probe attenuation (:CHANnel<n>:PROBe), and probe units (:CHANnel<n>:UNITs). Always set these values first before setting ABSolute thresholds

When <source> is not included in the command or query, the source identified by :MEASure:SOURce is used.

The ":MEASure:DEFine THResholds" command also lets you specify absolute measurement thresholds.

Query Syntax :MEASure:THResholds:ABSolute? [<source>]

The :MEASure:THResholds:ABSolute? query returns the upper, middle, and lower measurement threshold absolute values.

Return Format <upper>,<middle>,<lower><NL>
 <upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.

- See Also**
- [":MEASure:THResholds:METHod"](#) on page 747
 - [":MEASure:THResholds:PERCent"](#) on page 748
 - [":MEASure:THResholds:STANdard"](#) on page 749
 - [":CHANnel<n>:RANGe"](#) on page 374
 - [":CHANnel<n>:SCALE"](#) on page 375
 - [":CHANnel<n>:PROBe"](#) on page 356
 - [":CHANnel<n>:UNITs"](#) on page 376
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:DELay:DEFine"](#) on page 685

:MEASure:THResholds:METhod

N (see [page 1708](#))

Command Syntax :MEASure:THResholds:METhod <threshold_mode>[,<source>]
 <threshold_mode> ::= {PERCent | ABSolute}
 <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:THResholds:METhod command specifies the method for entering measurement threshold values:

- **PERCent** — Lets you set the measurement thresholds to user-defined percentages between 5% and 95% of values between Vbase and Vtop. See :MEASure:THResholds:PERCent.
- **ABSolute** — Lets you set the measurement thresholds to absolute values. ABSolute thresholds are dependent on channel scaling (:CHANnel<n>:RANGe or :CHANnel<n>:SCALE), probe attenuation (:CHANnel<n>:PROBe), and probe units (:CHANnel<n>:UNITs). Always set these values first before setting ABSolute thresholds. See :MEASure:THResholds:ABSolute.

When <source> is not included in the command or query, the source identified by :MEASure:SOURce is used.

The ":MEASure:DEFine THResholds" command also lets you specify percent or absolute measurement thresholds.

Query Syntax :MEASure:THResholds:METhod? [<source>]

The :MEASure:THResholds:METhod? query returns the selected measurement threshold entry method.

Return Format <threshold_mode><NL>
 <threshold_mode> ::= {PERC | ABS}

- See Also**
- [":MEASure:THResholds:ABSolute"](#) on page 746
 - [":MEASure:THResholds:PERCent"](#) on page 748
 - [":MEASure:THResholds:STANdard"](#) on page 749
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:DELay:DEFine"](#) on page 685

:MEASure:THResholds:PERCent

N (see [page 1708](#))

Command Syntax :MEASure:THResholds:PERCent <upper>,<middle>,<lower>[,<source>]
 <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> | 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:THResholds:PERCent command sets the upper, middle, and lower measurement thresholds to user-defined percentages between 5% and 95% of values between Vbase and Vtop.

When <source> is not included in the command or query, the source identified by :MEASure:SOURce is used.

The ":MEASure:DEFine THResholds" command also lets you specify percent measurement thresholds.

Query Syntax :MEASure:THResholds:PERCent? [<source>]

The :MEASure:THResholds:PERCent? query returns the upper, middle, and lower measurement threshold percent values.

Return Format <upper>,<middle>,<lower><NL>
 <upper>,<middle>,<lower> ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.

- See Also**
- [":MEASure:THResholds:ABSolute"](#) on page 746
 - [":MEASure:THResholds:METHod"](#) on page 747
 - [":MEASure:THResholds:STANdard"](#) on page 749
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:DELay:DEFine"](#) on page 685

:MEASure:THResholds:STANdard

N (see [page 1708](#))

Command Syntax :MEASure:THResholds:STANdard [<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:THResholds:STANdard command sets the lower, middle, and upper measurement thresholds to 10%, 50%, and 90% values between Vbase and Vtop.

When <source> is not included in the command, the source identified by :MEASure:SOURce is used.

The ":MEASure:DEFine THResholds" command also lets you set standard measurement thresholds.

- See Also**
- [":MEASure:THResholds:ABSolute"](#) on page 746
 - [":MEASure:THResholds:METHod"](#) on page 747
 - [":MEASure:THResholds:PERCent"](#) on page 748
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:DELay:DEFine"](#) on page 685

:MEASure:TIEClock

N (see [page 1708](#))

Command Syntax

NOTE

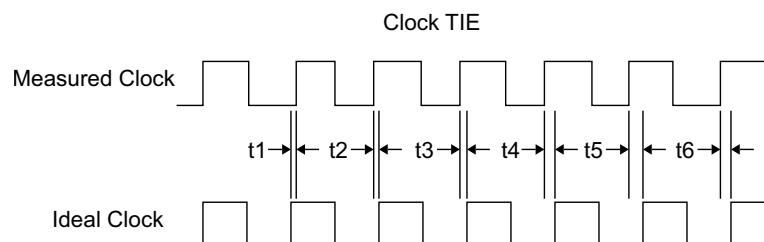
This measurement is available when the optional jitter analysis feature is licensed

```
:MEASure:TIEClock <config>
<config> ::= <source>, {SECond | UNITinterval}, <direction>
<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
<direction> ::= {RISing | FALLing | BOTH}
```

The :MEASure:TIEClock command installs a clock TIE jitter measurement on screen.

The Clock TIE (Time Interval Error) jitter measurement compares edges in a clock signal with the edges in an ideal clock signal determined by the clock recovery feature. To change the clock recovery method, see [":CLOCK:METHod"](#) on page 382.

Once the ideal clock frequency is determined, all of the cycles of your waveform are compared to the ideal clock frequency and the error statistics are computed. The following figure shows how the Clock TIE is measured on rising edges of the clock.



The measurement is made on all the cycles of the waveform. Even in Zoom mode, all the cycles of the waveform in the Main window will be measured.

Query Syntax

```
:MEASure:TIEClock? <config>
```

The :MEASure:TIEClock? query measures and returns the Clock TIE jitter value.

Return Format

```
<value><NL>
```

```
<value> ::= Clock TIE jitter in NR3 format
```

- See Also
- [":MEASure:TIEData"](#) on page 752
 - [Chapter 24](#), ":JITTer Commands," starting on page 593

:MEASure:TIEData

N (see [page 1708](#))

Command Syntax

NOTE

This measurement is available when the optional jitter analysis feature is licensed

```

:MEASure:TIEData <config>

<config> ::= <source>, {SECond | UNITinterval}

<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

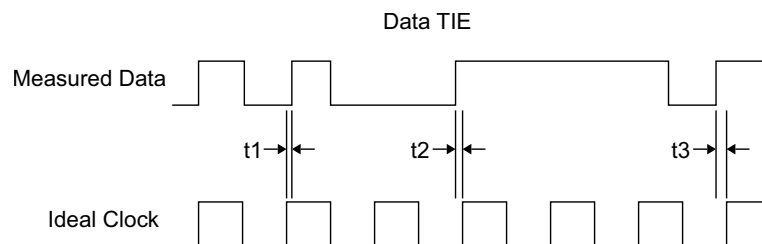
<r> ::= 1 to (# ref waveforms) in NR1 format

```

The :MEASure:TIEData command installs a data TIE jitter measurement on screen.

The Data TIE (Time Interval Error) jitter measurement compares edges in a data signal with the edges in an ideal data signal determined by the clock recovery feature. To change the clock recovery method, see **"[:CLOCK:METHod](#)"** on page 382.

Once the ideal data rate is determined, all of the data intervals are measured with respect to the ideal data rate and the error statistics are computed. The following figure shows how the Data TIE is measured on rising edges of the data.



The measurement is made on all the cycles of the waveform. Even in Zoom mode, all the cycles of the waveform in the Main window will be measured.

Query Syntax :MEASure:TIEData? <config>

The :MEASure:TIEData? query measures and returns the Data TIE jitter value.

Return Format

```

<value><NL>

<value> ::= Clock TIE jitter in NR3 format

```

- See Also
- **":MEASure:TIEClock"** on page 750
 - **Chapter 24**, "JITTer Commands," starting on page 593

:MEASure:TVALue?

C (see [page 1708](#))

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> | 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

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 671
 - [":MEASure:TEDGe"](#) on page 742
 - [":MEASure:YATX"](#) on page 768

:MEASure:VAMPlitude

C (see [page 1708](#))

Command Syntax :MEASure:VAMPlitude [<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: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:

$$\text{vertical amplitude} = V_{\text{top}} - V_{\text{base}}$$

Return Format <value><NL>
 <value> ::= the amplitude of the selected waveform in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VBASe"](#) on page 758
 - [":MEASure:VTOP"](#) on page 764
 - [":MEASure:VPP"](#) on page 761

:MEASure:VAVerage

C (see [page 1708](#))

Command Syntax :MEASure:VAVerage [<interval>][,<source>]

<interval> ::= {CYCLe | DISPlay}

<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: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 is specified, the current source is modified.

Query Syntax :MEASure:VAVerage? [<interval>][,<source>]

The :MEASure:VAVerage? query returns the average value of an integral number of periods of the signal.

Return Format <value><NL>

<value> ::= calculated average value in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:VBASe

C (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VTOP"](#) on page 764
 - [":MEASure:VAMPLitude"](#) on page 756
 - [":MEASure:VMIN"](#) on page 760

:MEASure:VMAX

C (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VMIN"](#) on page 760
 - [":MEASure:VPP"](#) on page 761
 - [":MEASure:VTOP"](#) on page 764

:MEASure:VMIN

C (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VBASe"](#) on page 758
 - [":MEASure:VMAX"](#) on page 759
 - [":MEASure:VPP"](#) on page 761

:MEASure:VPP

C (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VMAX"](#) on page 759
 - [":MEASure:VMIN"](#) on page 760
 - [":MEASure:VAMPLitude"](#) on page 756

:MEASure:VRATio

N (see [page 1708](#))

Command Syntax :MEASure:VRATio [<interval>] [,<source1>] [,<source2>]

<interval> ::= {CYCLe | DISPlay}

<source1,2> ::= {CHANnel<n> | FUNCTio<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 671
 - [":MEASure:VRMS"](#) on page 763
 - [":MEASure:SOURce"](#) on page 733

:MEASure:VRMS

C (see [page 1708](#))

Command Syntax :MEASure:VRMS [<interval>] [,<type>] [,<source>]
 <interval> ::= {CYCLe | DISPlay}
 <type> ::= {AC | DC}
 <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

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 671
- [":MEASure:SOURce"](#) on page 733

:MEASure:VTOP

C (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:VMAX"](#) on page 759
 - [":MEASure:VAMPLitude"](#) on page 756
 - [":MEASure:VBASe"](#) on page 758

:MEASure:WINDow

N (see [page 1708](#))

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 671
 - [":MEASure:SOURce"](#) on page 733

:MEASure:XMAX

N (see [page 1708](#))

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.

Return Format <value><NL>

<value> ::= horizontal value of the maximum in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:XMIN"](#) on page 767

:MEASure:XMIN

N (see [page 1708](#))

Command Syntax :MEASure:XMIN [<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: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.

Return Format <value><NL>

<value> ::= horizontal value of the minimum in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:XMAX"](#) on page 766

:MEASure:YATX

N (see [page 1708](#))

Command Syntax :MEASure:YATX <horiz_location>[,<source>]

<horiz_location> ::= time from trigger in seconds

<source> ::= {<digital channels> | CHANnel<n> | FUNction<m> | MATH<m>
| FFT | WMEMory<r>}

<digital channels> ::= DIGital<d>

<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:YATX command installs a screen measurement of 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 <horiz_location> is a frequency value instead of a time value.

Query Syntax :MEASure:YATX? <horiz_location>[,<source>]

The :MEASure:YATX? query returns the vertical value at a specified horizontal value on the source specified (see also :MEASure:SOURce).

Return Format <value><NL>

<value> ::= vertical value at the specified horizontal location
in NR3 format

- See Also**
- ["Introduction to :MEASure Commands"](#) on page 671
 - [":MEASure:SOURce"](#) on page 733
 - [":MEASure:TEDGe"](#) on page 742
 - [":MEASure:TVALue?"](#) on page 754

29 :MEASure Power Commands

These :MEASure commands are available when the power measurements and analysis application is licensed and enabled.

Table 111 :MEASure Power Commands Summary

Command	Query	Options and Query Returns
:MEASure:ANGLE [<source1>] [, <source2>] >] (see page 774)	:MEASure:ANGLE? [<source1>] [, <source2>] >] (see page 774)	<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 page 775)	:MEASure:APParent? [<source1>] [, <source2>] >] (see page 775)	<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 page 776)	:MEASure:CPLoss? [<source1>] [, <source2>] >] (see page 776)	<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 111 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:CRESt [<source>] (see page 777)	:MEASure:CRESt? [<source>] (see page 777)	<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 778)	:MEASure:EFFiciency? (see page 778)	<return_value> ::= percent value in NR3 format
:MEASure:ELOSs [<source>] (see page 779)	:MEASure:ELOSs? [<source>] (see page 779)	<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 780)	:MEASure:FACTOR? [<source1>] [,<source2 >] (see page 780)	<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 781)	:MEASure:IPower? (see page 781)	<return_value> ::= the input power value in NR3 format
:MEASure:OFFTime [<source1>] [,<source2 >] (see page 782)	:MEASure:OFFTime? [<source1>] [,<source2 >] (see page 782)	<source1>, <source2> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <return_value> ::= the time in seconds in NR3 format

Table 111 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:ONTime [<source1>] [, <source2>] >] (see page 783)	:MEASure:ONTime? [<source1>] [, <source2>] >] (see page 783)	<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 784)	:MEASure:OPower? (see page 784)	<return_value> ::= the output power value in NR3 format
:MEASure:PCURrent [<source>] (see page 785)	:MEASure:PCURrent? [<source>] (see page 785)	<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 page 786)	:MEASure:PLOss? [<source>] (see page 786)	<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 111 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:RDSon [<source1>] [, <source2>] (see page 787)	:MEASure:RDSon? [<source1>] [, <source2>] (see page 787)	<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 Rds(on) value in NR3 format
:MEASure:REACtive [<source1>] [, <source2>] (see page 788)	:MEASure:REACtive? [<source1>] [, <source2>] (see page 788)	<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 page 789)	:MEASure:REAL? [<source>] (see page 789)	<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 790)	:MEASure:RIPple? [<source>] (see page 790)	<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 111 :MEASure Power Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:TRESponse [<source>] (see page 791)	:MEASure:TRESponse? [<source>] (see page 791)	<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 792)	:MEASure:VCESat? [<source>] (see page 792)	<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 1708](#))

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 733
 - [":POWer:QUALity:APPLy"](#) on page 907

:MEASure:APParent

N (see [page 1708](#))

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 733
 - [":POWer:QUALity:APPLy"](#) on page 907

:MEASure:CPLoss

N (see [page 1708](#))

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 733
 - [":POWer:SWITCh:APPLy"](#) on page 930

:MEASure:CRESt

N (see [page 1708](#))

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 733
 - [":POWer:QUALity:APPLy"](#) on page 907

:MEASure:EFFiciency

N (see [page 1708](#))

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 927
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:EFFiciency:APPLY"](#) on page 864

:MEASure:ELOSs

N (see [page 1708](#))

Command Syntax :MEASure:ELOSs [<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:ELOSs 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 733
 - [":POWer:SWITCh:APPLY"](#) on page 930

:MEASure:FACTOR

N (see [page 1708](#))

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 733
 - [":POWer:QUALity:APPLY"](#) on page 907

:MEASure:IPower

N (see [page 1708](#))

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 927
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:EFFiciency:APPLY"](#) on page 864

:MEASure:OFFTime

N (see [page 1708](#))

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 733
 - [":POWer:ONOff:TEST"](#) on page 890
 - [":POWer:ONOff:APPLy"](#) on page 887

:MEASure:ONTime

N (see [page 1708](#))

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 733
 - [":POWer:ONOff:TEST"](#) on page 890
 - [":POWer:ONOff:APPLY"](#) on page 887

:MEASure:OPower

N (see [page 1708](#))

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 927
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:EFFiciency:APPLY"](#) on page 864

:MEASure:PCURrent

N (see [page 1708](#))

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 733
 - [":POWer:INRush:APPLy"](#) on page 879

:MEASure:PLOSs

N (see [page 1708](#))

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 733
 - [":POWER:SWITCh:APPLY"](#) on page 930

:MEASure:RDson

N (see [page 1708](#))

Command Syntax :MEASure:RDson [<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: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 Rds(on) value in NR3 format

See Also • [":MEASure:VCESat"](#) on page 792

:MEASure:REActive

N (see [page 1708](#))

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 733
 - [":POWER:QUALity:APPLy"](#) on page 907

:MEASure:REAL

N (see [page 1708](#))

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 733
 - [":POWER:QUALity:APPLy"](#) on page 907

:MEASure:RIPple

N (see [page 1708](#))

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: $V_{max} - V_{min}$.

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 733
 - [":POWer:RIPple:APPLY"](#) on page 908

:MEASure:TRESponse

N (see [page 1708](#))

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 733
 - [":POWER:TRANSient:APPLY"](#) on page 936

:MEASure:VCESat

N (see [page 1708](#))

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 787

30 :MTESt Commands

The MTESt subsystem commands and queries control the mask test features. See ["Introduction to :MTESt Commands"](#) on page 795.

Table 112 :MTESt Commands Summary

Command	Query	Options and Query Returns
:MTESt:ALL { {0 OFF} {1 ON} } (see page 799)	:MTESt:ALL? (see page 799)	{0 1}
:MTESt:AMASk:CREate (see page 800)	n/a	n/a
:MTESt:AMASk:SOURce <source> (see page 801)	:MTESt:AMASk:SOURce? (see page 801)	<source> ::= CHANnel<n> <n> ::= {1 2 3 4} for 4ch models <n> ::= {1 2} for 2ch models
:MTESt:AMASk:UNITs <units> (see page 802)	:MTESt:AMASk:UNITs? (see page 802)	<units> ::= {CURRent DIVisions}
:MTESt:AMASk:XDELta <value> (see page 803)	:MTESt:AMASk:XDELta? (see page 803)	<value> ::= X delta value in NR3 format
:MTESt:AMASk:YDELta <value> (see page 804)	:MTESt:AMASk:YDELta? (see page 804)	<value> ::= Y delta value in NR3 format
n/a	:MTESt:COUNT:FWAVEfor ms? [CHANnel<n>] (see page 805)	<failed> ::= number of failed waveforms in NR1 format
:MTESt:COUNT:RESet (see page 806)	n/a	n/a
n/a	:MTESt:COUNT:TIME? (see page 807)	<time> ::= elapsed seconds in NR3 format
n/a	:MTESt:COUNT:WAVEform s? (see page 808)	<count> ::= number of waveforms in NR1 format
:MTESt:DATA <mask> (see page 809)	:MTESt:DATA? (see page 809)	<mask> ::= data in IEEE 488.2 # format.

Table 112 :MTESt Commands Summary (continued)

Command	Query	Options and Query Returns
:MTESt:DELeTe (see page 810)	n/a	n/a
:MTESt:ENABLe {{0 OFF} {1 ON}} (see page 811)	:MTESt:ENABLe? (see page 811)	{0 1}
:MTESt:LOCK {{0 OFF} {1 ON}} (see page 812)	:MTESt:LOCK? (see page 812)	{0 1}
:MTESt:RMODE <rmode> (see page 813)	:MTESt:RMODE? (see page 813)	<rmode> ::= {FORever TIME SIGMa WAVEforms}
:MTESt:RMODE:FACTION:MEASure {{0 OFF} {1 ON}} (see page 814)	:MTESt:RMODE:FACTION:MEASure? (see page 814)	{0 1}
:MTESt:RMODE:FACTION:PRINT {{0 OFF} {1 ON}} (see page 815)	:MTESt:RMODE:FACTION:PRINT? (see page 815)	{0 1}
:MTESt:RMODE:FACTION:SAVE {{0 OFF} {1 ON}} (see page 816)	:MTESt:RMODE:FACTION:SAVE? (see page 816)	{0 1}
:MTESt:RMODE:FACTION:STOP {{0 OFF} {1 ON}} (see page 817)	:MTESt:RMODE:FACTION:STOP? (see page 817)	{0 1}
:MTESt:RMODE:SIGMa <level> (see page 818)	:MTESt:RMODE:SIGMa? (see page 818)	<level> ::= from 0.1 to 9.3 in NR3 format
:MTESt:RMODE:TIME <seconds> (see page 819)	:MTESt:RMODE:TIME? (see page 819)	<seconds> ::= from 1 to 86400 in NR3 format
:MTESt:RMODE:WAVEform s <count> (see page 820)	:MTESt:RMODE:WAVEform s? (see page 820)	<count> ::= number of waveforms in NR1 format
:MTESt:SCALE:BIND {{0 OFF} {1 ON}} (see page 821)	:MTESt:SCALE:BIND? (see page 821)	{0 1}
:MTESt:SCALE:X1 <x1_value> (see page 822)	:MTESt:SCALE:X1? (see page 822)	<x1_value> ::= X1 value in NR3 format
:MTESt:SCALE:XDELta <xdelta_value> (see page 823)	:MTESt:SCALE:XDELta? (see page 823)	<xdelta_value> ::= X delta value in NR3 format

Table 112 :MTESt Commands Summary (continued)

Command	Query	Options and Query Returns
:MTESt:SCALe:Y1 <y1_value> (see page 824)	:MTESt:SCALe:Y1? (see page 824)	<y1_value> ::= Y1 value in NR3 format
:MTESt:SCALe:Y2 <y2_value> (see page 825)	:MTESt:SCALe:Y2? (see page 825)	<y2_value> ::= Y2 value in NR3 format
:MTESt:SOURce <source> (see page 826)	:MTESt:SOURce? (see page 826)	<source> ::= {CHANnel<n> NONE} <n> ::= {1 2 3 4} for 4ch models <n> ::= {1 2} for 2ch models
n/a	:MTESt:TITLe? (see page 827)	<title> ::= a quoted 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 +2.50000000E-01;YDEL +2.50000000E-01;: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 +1.000E+03;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
```

```
#If VBA7 Then
```

```
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
```

```

#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 10000 ' Set I/O communication timeout.

    ' Make sure oscilloscope is running.
    myScope.WriteString ":RUN"

    ' Enable mask testing.
    myScope.WriteString ":MTESt:ENABle ON"

    ' 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 ":STATUS:OPERation:MTESt:ENABLE " + CStr(CInt("&H4
00"))

' 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 ":STATUS:OPERation: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 ":STATUS:OPERation: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 1708](#))

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:ALL?

The :MTESt:ALL? query returns the current setting.

Return Format <on_off><NL>

<on_off> ::= {1 | 0}

See Also • ["Introduction to :MTESt Commands"](#) on page 795

:MTESt:AMASk:CREate

N (see [page 1708](#))

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 795
 - [":MTESt:AMASk:XDELta"](#) on page 803
 - [":MTESt:AMASk:YDELta"](#) on page 804
 - [":MTESt:AMASk:UNITs"](#) on page 802
 - [":MTESt:AMASk:SOURce"](#) on page 801
 - [":MTESt:SOURce"](#) on page 826

- Example Code**
- ["Example Code"](#) on page 795

:MTESt:AMASk:SOURce

N (see [page 1708](#))

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 795
 - [":MTESt:AMASk:XDELta"](#) on page 803
 - [":MTESt:AMASk:YDELta"](#) on page 804
 - [":MTESt:AMASk:UNITs"](#) on page 802
 - [":MTESt:SOURce"](#) on page 826

Example Code • ["Example Code"](#) on page 795

:MTESt:AMASk:UNITs

N (see [page 1708](#))

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 ΔX and voltage for ΔY .
- **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 795
 - [":MTESt:AMASk:XDELta"](#) on page 803
 - [":MTESt:AMASk:YDELta"](#) on page 804
 - [":CHANnel<n>:UNITs"](#) on page 376
 - [":MTESt:AMASk:SOURce"](#) on page 801
 - [":MTESt:SOURce"](#) on page 826

Example Code • ["Example Code"](#) on page 795

:MTESt:AMASk:XDELta

N (see [page 1708](#))

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 ± 250 ms. If the setting for :MTESt:AMASk:UNITs is DIVisions, the same X delta value will set the tolerance to ± 250 millidivisions, or 1/4 of a division.

Query Syntax :MTESt:AMASk:XDELta?

The :MTESt:AMASk:XDELta? query returns the current setting of the ΔX 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 795
 - [":MTESt:AMASk:UNITs"](#) on page 802
 - [":MTESt:AMASk:YDELta"](#) on page 804
 - [":MTESt:AMASk:SOURce"](#) on page 801
 - [":MTESt:SOURce"](#) on page 826

Example Code • ["Example Code"](#) on page 795

:MTESt:AMASk:YDELta

N (see [page 1708](#))

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 ± 250 mV. If the setting for :MTESt:AMASk:UNITs is DIVisions, the same Y delta value will set the tolerance to ± 250 millidivisions, or 1/4 of a division.

Query Syntax :MTESt:AMASk:YDELta?

The :MTESt:AMASk:YDELta? query returns the current setting of the ΔY 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 795
 - [":MTESt:AMASk:UNITs"](#) on page 802
 - [":MTESt:AMASk:XDELta"](#) on page 803
 - [":MTESt:AMASk:SOURce"](#) on page 801
 - [":MTESt:SOURce"](#) on page 826

Example Code • ["Example Code"](#) on page 795

:MTESt:COUNT:FWAVEforms?

N (see [page 1708](#))

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 795
 - [":MTESt:COUNT:WAVEforms?"](#) on page 808
 - [":MTESt:COUNT:TIME?"](#) on page 807
 - [":MTESt:COUNT:RESet"](#) on page 806
 - [":MTESt:SOURce"](#) on page 826

Example Code • ["Example Code"](#) on page 795

:MTESt:COUNT:RESet

N (see [page 1708](#))

Command Syntax :MTESt:COUNT:RESet

The :MTESt:COUNT:RESet command resets the mask statistics.

- See Also**
- ["Introduction to :MTESt Commands"](#) on page 795
 - [":MTESt:COUNT:WAVEforms?"](#) on page 808
 - [":MTESt:COUNT:FWAVEforms?"](#) on page 805
 - [":MTESt:COUNT:TIME?"](#) on page 807

:MTESt:COUNT:TIME?

N (see [page 1708](#))

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 795
 - [":MTESt:COUNT:WAVEforms?"](#) on page 808
 - [":MTESt:COUNT:FWAVEforms?"](#) on page 805
 - [":MTESt:COUNT:RESet"](#) on page 806

Example Code • ["Example Code"](#) on page 795

:MTESt:COUNT:WAVEforms?

N (see [page 1708](#))

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 795
 - [":MTESt:COUNT:FWAVEforms?"](#) on page 805
 - [":MTESt:COUNT:TIME?"](#) on page 807
 - [":MTESt:COUNT:RESet"](#) on page 806

Example Code • ["Example Code"](#) on page 795

:MTESt:DATA

N (see [page 1708](#))

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 963
 - [":RECall:MASK\[:START\]"](#) on page 948

:MTESt:DELeTe

N (see [page 1708](#))

Command Syntax :MTESt:DELeTe

The :MTESt:DELeTe command clears the currently loaded mask.

- See Also**
- ["Introduction to :MTESt Commands"](#) on page 795
 - [":MTESt:AMASk:CREate"](#) on page 800

:MTESt:ENABle

N (see [page 1708](#))

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 795

:MTESt:LOCK

N (see [page 1708](#))

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 795
 - [":MTESt:SOURce"](#) on page 826

:MTESt:RMODe

N (see [page 1708](#))

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 818 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 819 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 820 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 795
 - [":MTESt:RMODe:SIGMa"](#) on page 818
 - [":MTESt:RMODe:TIME"](#) on page 819
 - [":MTESt:RMODe:WAVeforms"](#) on page 820

Example Code • ["Example Code"](#) on page 795

:MTESt:RMODe:FACTion:MEASure

N (see [page 1708](#))

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 795
 - [":MTESt:RMODe:FACTion:PRINT"](#) on page 815
 - [":MTESt:RMODe:FACTion:SAVE"](#) on page 816
 - [":MTESt:RMODe:FACTion:STOP"](#) on page 817

:MTESt:RMODE:FACTION:PRINT

N (see [page 1708](#))

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 22](#), “:HCOPY Commands,” starting on page 573 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 795
 - [":MTESt:RMODE:FACTION:MEASure"](#) on page 814
 - [":MTESt:RMODE:FACTION:SAVE"](#) on page 816
 - [":MTESt:RMODE:FACTION:STOP"](#) on page 817

:MTESt:RMODe:FACTion:SAVE

N (see [page 1708](#))

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 34](#), “:SAVE Commands,” starting on page 953 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 795
 - [":MTESt:RMODe:FACTion:MEASure"](#) on page 814
 - [":MTESt:RMODe:FACTion:PRINT"](#) on page 815
 - [":MTESt:RMODe:FACTion:STOP"](#) on page 817

:MTESt:RMODe:FACTion:STOP

N (see [page 1708](#))

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 795
 - [":MTESt:RMODe:FACTion:MEASure"](#) on page 814
 - [":MTESt:RMODe:FACTion:PRINT"](#) on page 815
 - [":MTESt:RMODe:FACTion:SAVE"](#) on page 816

:MTESt:RMODe:SIGMa

N (see [page 1708](#))

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 795
- [":MTESt:RMODe"](#) on page 813

Example Code

- ["Example Code"](#) on page 795

:MTESt:RMODE:TIME

N (see [page 1708](#))

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 795
- [":MTESt:RMODE"](#) on page 813

:MTESt:RMODE:WAVEforms

N (see [page 1708](#))

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 795
 - [":MTESt:RMODE"](#) on page 813

:MTESt:SCALE:BIND

N (see [page 1708](#))

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 795
 - [":MTESt:SCALE:X1"](#) on page 822
 - [":MTESt:SCALE:XDELta"](#) on page 823
 - [":MTESt:SCALE:Y1"](#) on page 824
 - [":MTESt:SCALE:Y2"](#) on page 825

:MTESt:SCALe:X1

N (see [page 1708](#))

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 795
 - [":MTESt:SCALe:BIND"](#) on page 821
 - [":MTESt:SCALe:XDELta"](#) on page 823
 - [":MTESt:SCALe:Y1"](#) on page 824
 - [":MTESt:SCALe:Y2"](#) on page 825

:MTESt:SCALe:XDELta

N (see [page 1708](#))

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 ΔX , redefining ΔX also moves those vertices to stay in the same locations with respect to X1 and ΔX . 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 ΔX .

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 ΔX 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 ΔX .

Return Format <xdelta_value><NL>

<xdelta_value> ::= X delta value in NR3 format

- See Also**
- ["Introduction to :MTESt Commands"](#) on page 795
 - [":MTESt:SCALe:BIND"](#) on page 821
 - [":MTESt:SCALe:X1"](#) on page 822
 - [":MTESt:SCALe:Y1"](#) on page 824
 - [":MTESt:SCALe:Y2"](#) on page 825

:MTESt:SCALe:Y1

N (see [page 1708](#))

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 795
 - [":MTESt:SCALe:BIND"](#) on page 821
 - [":MTESt:SCALe:X1"](#) on page 822
 - [":MTESt:SCALe:XDELta"](#) on page 823
 - [":MTESt:SCALe:Y2"](#) on page 825

:MTESt:SCALe:Y2

N (see [page 1708](#))

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 795
 - [":MTESt:SCALe:BIND"](#) on page 821
 - [":MTESt:SCALe:X1"](#) on page 822
 - [":MTESt:SCALe:XDELta"](#) on page 823
 - [":MTESt:SCALe:Y1"](#) on page 824

:MTESt:SOURce

N (see [page 1708](#))

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 795
 - [":MTESt:AMASK:SOURce"](#) on page 801

:MTESt:TITLe?

N (see [page 1708](#))

Query Syntax :MTESt:TITLe?

The :MTESt:TITLe? query returns the mask title which is a quoted 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 quoted string of up to 128 ASCII characters.

See Also • ["Introduction to :MTESt Commands"](#) on page 795

31 :POD Commands

Control all oscilloscope functions associated with groups of digital channels. See ["Introduction to :POD<n> Commands"](#) on page 829.

Table 113 :POD<n> Commands Summary

Command	Query	Options and Query Returns
:POD<n>:DISPlay { {0 OFF} {1 ON} } (see page 831)	:POD<n>:DISPlay? (see page 831)	{0 1} <n> ::= 1-2 in NR1 format
:POD<n>:NIBBle<n>:THR eshold <value> (see page 832)	:POD<n>:NIBBle<n>:THR eshold? (see page 832)	<n> ::= 1-2 in NR1 format <value> ::= {CMOS ECL TTL <decimal>[<suffix>]} <decimal> ::= -8.00 to +8.00 in NR3 format [suffix] ::= {V mV uV }
:POD<n>:SIZE <value> (see page 834)	:POD<n>:SIZE? (see page 834)	<value> ::= {SMALl MEDium LARGe}
:POD<n>:THR eshold <value> (see page 835)	:POD<n>:THR eshold? (see page 835)	<n> ::= 1-2 in NR1 format <value> ::= {CMOS ECL TTL <decimal>[<suffix>]} <decimal> ::= -8.00 to +8.00 in NR3 format [suffix] ::= {V mV uV }

Introduction to <n> ::= {1 | 2}

:POD<n>
Commands

The POD subsystem commands control the viewing and threshold of groups of digital channels.

POD1 ::= D0-D7

POD2 ::= D8-D15

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 1708](#))

Command Syntax :POD<n>:DISPlay <display>

<display> ::= {{1 | ON} | {0 | OFF}}

<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

The :POD<n>:DISPlay command turns displaying of the specified group of channels on or off.

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 829
 - [":DIGital<d>:DISPlay"](#) on page 400
 - [":CHANnel<n>:DISPlay"](#) on page 351
 - [":VIEW"](#) on page 281
 - [":BLANK"](#) on page 270
 - [":STATus?"](#) on page 278

:POD<n>:NIBBle<n>:THReshold

N (see [page 1708](#))

Command Syntax :POD<n>:NIBBle<n>:THReshold <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.

<value> ::= {CMOS | ECL | TTL | <decimal>[<suffix>]}

<decimal> ::= -8.00 to +8.00 in NR3 format

<suffix> ::= {V | mV | uV}

POD1 ::= D0-D7

POD2 ::= D8-D15

The :POD<n>:NIBBle<n>:THReshold command sets the logic threshold value for a nibble within a pod:

- Pod 1 has the digital channels 0 through 7.
 - Pod 1, nibble 1 has the digital channels 0 through 3.
 - Pod 1, nibble 2 has the digital channels 4 through 7.
- Pod 2 has the digital channels 8 through 15.
 - Pod 2, nibble 1 has the digital channels 8 through 11.
 - Pod 2, nibble 2 has the digital channels 12 through 15.

	POD2								POD1							
	NIBBle2				NIBBle1				NIBBle2				NIBBle1			
Digital channel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The threshold is used for triggering purposes and for displaying the digital data as high (above the threshold) or low (below the threshold).

The voltage values for the predefined thresholds are:

- CMOS = 2.5 V
- ECL = -1.3 V
- TTL = 1.4 V

Query Syntax :POD<n>:NIBBle<n>:THReshold?

The :POD<N>:NIBBle<N>:THReshold? query returns the threshold value for the specified pod and nibble.

Return Format <threshold><NL>

<threshold> ::= Floating point number in NR3 format

- See Also**
- ["Introduction to :POD<n> Commands"](#) on page 829
 - [":POD<n>:THReshold"](#) on page 835
 - [":DIGital<d>:THReshold"](#) on page 404
 - [":TRIGger\[:EDGE\]:LEVel"](#) on page 1507

Example Code This example sets the threshold to 1.8 volts for pod 2, nibble 2 (digital channels D15 through D12).

```
myScope.WriteString ":POD2:NIBBle2:THReshold 1.8"
```

:POD<n>:SIZE

N (see [page 1708](#))

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.

Query Syntax :POD<n>:SIZE?

The :POD<n>:SIZE? query returns the digital channels size setting.

Return Format <size_value><NL>

<size_value> ::= {SMALL | MEDIUM | LARGE}

- See Also**
- ["Introduction to :POD<n> Commands"](#) on page 829
 - [":DIGital<d>:SIZE"](#) on page 403
 - [":DIGital:ORDER\[:SET\]"](#) on page 399

:POD<n>:THReshold

N (see [page 1708](#))

Command Syntax :POD<n>:THReshold <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.

<value> ::= {CMOS | ECL | TTL | <decimal>[<suffix>]}

<decimal> ::= -8.00 to +8.00 in NR3 format

<suffix> ::= {V | mV | uV}

POD1 ::= D0-D7

POD2 ::= D8-D15

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).

The voltage values for the predefined thresholds are:

- CMOS = 2.5 V
- ECL = -1.3 V
- TTL = 1.4 V

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 829
 - [":DIGital<d>:THReshold"](#) on page 404
 - [":TRIGger\[:EDGE\]:LEVel"](#) on page 1507

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 49](#), “Programming Examples,” starting on page 1717

32 :POWer Commands

These :POWer commands are available when the power measurements and analysis application is licensed and enabled.

Table 114 :POWer Commands Summary

Command	Query	Options and Query Returns
n/a	:POWer:CLResponse? (see page 845)	n/a
:POWer:CLResponse:APPLy (see page 846)	n/a	n/a
n/a	:POWer:CLResponse:DATA? [SWEep SINGle] (see page 847)	<binary_block> ::= comma-separated data with newlines at the end of each row
n/a	:POWer:CLResponse:DATA:GMARgin? (see page 848)	<gain_margin> ::= gain margin in dB in NR3 format.
n/a	:POWer:CLResponse:DATA:GMARgin:FREQuency? (see page 849)	<frequency> ::= 0 degrees phase crossover frequency in Hz in NR3 format
n/a	:POWer:CLResponse:DATA:PMARgin? (see page 850)	<phase_margin> ::= phase margin in degrees in NR3 format.
n/a	:POWer:CLResponse:DATA:PMARgin:FREQuency? (see page 851)	<frequency> ::= 0dB gain crossover frequency in Hz in NR3 format.
:POWer:CLResponse:FREQuency:MODE <mode> (see page 852)	:POWer:CLResponse:FREQuency:MODE? (see page 852)	<mode> ::= {SWEep SINGle}
:POWer:CLResponse:FREQuency:SINGle <value>[suffix] (see page 853)	:POWer:CLResponse:FREQuency:SINGle? (see page 853)	<value> ::= {20 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:CLResponse:FREQuency:STARt <value>[suffix] (see page 854)	:POWer:CLResponse:FREQuency:STARt? (see page 854)	<value> ::= {20 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:POWer:CLResponse:FREQuency:STOP <value>[suffix] (see page 855)	:POWer:CLResponse:FREQuency:STOP? (see page 855)	<value> ::= {100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWer:CLResponse:PPDeade <pts> (see page 856)	:POWer:CLResponse:PPDeade? (see page 856)	<pts> ::= {10 20 30 40 50 60 70 80 90 100}
:POWer:CLResponse:SOURce:INPut <source> (see page 857)	:POWer:CLResponse:SOURce:INPut? (see page 857)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:CLResponse:SOURce:OUTPut <source> (see page 858)	:POWer:CLResponse:SOURce:OUTPut? (see page 858)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:CLResponse:TRACe <selection> (see page 859)	:POWer:CLResponse:TRACe? (see page 859)	<selection> ::= {NONE ALL GAIN PHASe}[, {GAIN PHASe}]
:POWer:CLResponse:WGEN:LOAD <impedance> (see page 860)	:POWer:CLResponse:WGEN:LOAD? (see page 860)	<impedance> ::= {ONEMeg FIFTy}
:POWer:CLResponse:WGEN:VOLTagE <amplitude>[, <range>] (see page 861)	:POWer:CLResponse:WGEN:VOLTagE? [<range>] (see page 861)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F10MILLIHZ F100MILLIHZ F1HZ {F10HZ F20HZ} F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ {F20MHZ F100MHZ}}
:POWer:CLResponse:WGEN:VOLTagE:PROFile {0 OFF} {1 ON} (see page 862)	:POWer:CLResponse:WGEN:VOLTagE:PROFile? (see page 862)	{0 1}
:POWer:DESKew (see page 863)	n/a	n/a
:POWer:EFFiciency:APPLy (see page 864)	n/a	n/a

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:EFFiciency:TYPE <type> (see page 865)	:POWer:EFFiciency:TYPE? (see page 865)	<type> ::= {DCDC DCAC ACDC ACAC}
:POWer:ENABLE {{0 OFF} {1 ON}} (see page 866)	:POWer:ENABLE? (see page 866)	{0 1}
:POWer:HARMonics:APPLY (see page 867)	n/a	n/a
n/a	:POWer:HARMonics:DATA? (see page 868)	<binary_block> ::= comma-separated data with newlines at the end of each row
:POWer:HARMonics:DISPLAY <display> (see page 869)	:POWer:HARMonics:DISPLAY? (see page 869)	<display> ::= {TABLE BAR OFF}
n/a	:POWer:HARMonics:FAILCOUNT? (see page 870)	<count> ::= integer in NR1 format
:POWer:HARMonics:LINE <frequency> (see page 871)	:POWer:HARMonics:LINE? (see page 871)	<frequency> ::= {F50 F60 F400 AUTO}
n/a	:POWer:HARMonics:POWERFACTOR? (see page 872)	<value> ::= Class C power factor in NR3 format
:POWer:HARMonics:RPOWER <source> (see page 873)	:POWer:HARMonics:RPOWER? (see page 873)	<source> ::= {MEASured USER}
:POWer:HARMonics:RPOWER:USER <value> (see page 874)	:POWer:HARMonics:RPOWER:USER? (see page 874)	<value> ::= Watts from 1.0 to 600.0 in NR3 format
n/a	:POWer:HARMonics:RUNCOUNT? (see page 875)	<count> ::= integer in NR1 format
:POWer:HARMonics:STANDARD <class> (see page 876)	:POWer:HARMonics:STANDARD? (see page 876)	<class> ::= {A B C D}
n/a	:POWer:HARMonics:STATUS? (see page 877)	<status> ::= {PASS FAIL UNTested}
n/a	:POWer:HARMonics:THD? (see page 878)	<value> ::= Total Harmonics Distortion in NR3 format
:POWer:INRush:APPLY (see page 879)	n/a	n/a

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:INRush:EXIT (see page 880)	n/a	n/a
:POWer:INRush:NEXT (see page 881)	n/a	n/a
:POWer:ITYPE <type> (see page 882)	:POWer:ITYPE? (see page 882)	<type> ::= {DC AC}
:POWer:MODulation:APPLy (see page 883)	n/a	n/a
:POWer:MODulation:SOURce <source> (see page 884)	:POWer:MODulation:SOURce? (see page 884)	<source> ::= {V I}
:POWer:MODulation:TYPE <modulation> (see page 885)	:POWer:MODulation:TYPE? (see page 885)	<modulation> ::= {VAVerage ACRMs VRATio PERiod FREQuency PWIDith NWIDth DUTYcycle NDUTYcycle RISetime FALLtime}
:POWer:ONOff:APPLy (see page 887)	n/a	n/a
:POWer:ONOff:EXIT (see page 888)	n/a	n/a
:POWer:ONOff:NEXT (see page 889)	n/a	n/a
:POWer:ONOff:TEST { {0 OFF} {1 ON} } (see page 890)	:POWer:ONOff:TEST? (see page 890)	{0 1}
:POWer:ONOff:THResholDs <type>, <input_thr>, <output_thr> (see page 891)	:POWer:ONOff:THResholDs? <type> (see page 891)	<type> ::= {0 1} <input_thr> ::= percent from 0-100 in NR1 format <output_thr> ::= percent from 0-100 in NR1 format
n/a	:POWer:PSRR? (see page 893)	n/a
:POWer:PSRR:APPLy (see page 894)	n/a	n/a
n/a	:POWer:PSRR:DATA? [SWEp SINGle] (see page 895)	<binary_block> ::= comma-separated data with newlines at the end of each row

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:PSRR:FREQuency :MAXimum <value>[suffix] (see page 896)	:POWer:PSRR:FREQuency :MAXimum? (see page 896)	<value> ::= {10 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:FREQuency :MINimum <value>[suffix] (see page 897)	:POWer:PSRR:FREQuency :MINimum? (see page 897)	<value> ::= {1 10 100 1000 10000 100000 1000000 10000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:FREQuency :MODE <mode> (see page 898)	:POWer:PSRR:FREQuency :MODE? (see page 898)	<mode> ::= {SWEep SINGle}
:POWer:PSRR:FREQuency :SINGle <value>[suffix] (see page 899)	:POWer:PSRR:FREQuency :SINGle? (see page 899)	<value> ::= {1 10 100 1000 10000 100000 1000000 10000000 20000000} [suffix] ::= {Hz kHz MHz}
:POWer:PSRR:PPDecade <pts> (see page 900)	:POWer:PSRR:PPDecade? (see page 900)	<pts> ::= {10 20 30 40 50 60 70 80 90 100}
:POWer:PSRR:SOURce:IN Put <source> (see page 901)	:POWer:PSRR:SOURce:IN Put? (see page 901)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:PSRR:SOURce:OU TPut <source> (see page 902)	:POWer:PSRR:SOURce:OU TPut? (see page 902)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:PSRR:TRACe <selection> (see page 903)	:POWer:PSRR:TRACe? (see page 903)	<selection> ::= {NONE GAIN}
:POWer:PSRR:WGEN:LOAD <impedance> (see page 904)	:POWer:PSRR:WGEN:LOAD ? (see page 904)	<impedance> ::= {ONEMeg FIFTy}
:POWer:PSRR:WGEN:VOLT age <amplitude>[,<range>] (see page 905)	:POWer:PSRR:WGEN:VOLT age? [<range>] (see page 905)	<amplitude> ::= amplitude in volts in NR3 format <range> ::= {F10MILLIHZ F100MILLIHZ F1HZ {F10HZ F20HZ} F100HZ F1KHZ F10KHZ F100KHZ F1MHZ F10MHZ {F20MHZ F100MHZ}}
:POWer:PSRR:WGEN:VOLT age:PROFile {{0 OFF} {1 ON}} (see page 906)	:POWer:PSRR:WGEN:VOLT age:PROFile? (see page 906)	{0 1}

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:QUALity:APPLY (see page 907)	n/a	n/a
:POWer:RIPple:APPLY (see page 908)	n/a	n/a
:POWer:SIGNALs:AUTosetup <analysis> (see page 909)	n/a	<analysis> ::= {HARMONics EFFiciency RIPple MODulation QUALity SLEW SWITCH RDSVce}
:POWer:SIGNALs:CYCLes:HARMONics <count> (see page 910)	:POWer:SIGNALs:CYCLes:HARMONics? (see page 910)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWer:SIGNALs:CYCLes:QUALity <count> (see page 911)	:POWer:SIGNALs:CYCLes:QUALity? (see page 911)	<count> ::= integer in NR1 format Legal values are 1 to 100.
:POWer:SIGNALs:DURatiOn:EFFiciency <value>[suffix] (see page 912)	:POWer:SIGNALs:DURatiOn:EFFiciency? (see page 912)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:DURatiOn:MODulation <value>[suffix] (see page 913)	:POWer:SIGNALs:DURatiOn:MODulation? (see page 913)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:DURatiOn:ONOff:OFF <value>[suffix] (see page 914)	:POWer:SIGNALs:DURatiOn:ONOff:OFF? (see page 914)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:DURatiOn:ONOff:ON <value>[suffix] (see page 915)	:POWer:SIGNALs:DURatiOn:ONOff:ON? (see page 915)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:DURatiOn:RIPple <value>[suffix] (see page 916)	:POWer:SIGNALs:DURatiOn:RIPple? (see page 916)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:DURatiOn:TRANSient <value>[suffix] (see page 917)	:POWer:SIGNALs:DURatiOn:TRANSient? (see page 917)	<value> ::= value in NR3 format [suffix] ::= {s ms us ns}
:POWer:SIGNALs:IEXPeCted <value>[suffix] (see page 918)	:POWer:SIGNALs:IEXPeCted? (see page 918)	<value> ::= Expected current value in NR3 format [suffix] ::= {A mA}

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:SIGNals:OVERsh oot <percent> (see page 919)	:POWer:SIGNals:OVERsh oot? (see page 919)	<percent> ::= percent of overshoot value in NR1 format [suffix] ::= {V mV}
:POWer:SIGNals:VMAXim um:INRush <value>[suffix] (see page 920)	:POWer:SIGNals:VMAXim um:INRush? (see page 920)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:VMAXim um:ONOff:OFF <value>[suffix] (see page 921)	:POWer:SIGNals:VMAXim um:ONOff:OFF? (see page 921)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:VMAXim um:ONOff:ON <value>[suffix] (see page 922)	:POWer:SIGNals:VMAXim um:ONOff:ON? (see page 922)	<value> ::= Maximum expected input Voltage in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:VSTead y:ONOff:OFF <value>[suffix] (see page 923)	:POWer:SIGNals:VSTead y:ONOff:OFF? (see page 923)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:VSTead y:ONOff:ON <value>[suffix] (see page 924)	:POWer:SIGNals:VSTead y:ONOff:ON? (see page 924)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:VSTead y:TRANSient <value>[suffix] (see page 925)	:POWer:SIGNals:VSTead y:TRANSient? (see page 925)	<value> ::= Expected steady stage output Voltage value in NR3 format [suffix] ::= {V mV}
:POWer:SIGNals:SOURce :CURRent<i> <source> (see page 926)	:POWer:SIGNals:SOURce :CURRent<i>? (see page 926)	<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 927)	:POWer:SIGNals:SOURce :VOLTage<i>? (see page 927)	<i> ::= 1, 2 in NR1 format <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:POWer:SLEW:APPLy (see page 928)	n/a	n/a

Table 114 :POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:POWer:SLEW:SOURce <source> (see page 929)	:POWer:SLEW:SOURce? (see page 929)	<source> ::= {V I}
:POWer:SWITCh:APPLy (see page 930)	n/a	n/a
:POWer:SWITCh:CONduct ion <conduction> (see page 931)	:POWer:SWITCh:CONduct ion? (see page 931)	<conduction> ::= {WAVEform RDS VCE}
:POWer:SWITCh:IREFere nce <percent> (see page 932)	:POWer:SWITCh:IREFere nce? (see page 932)	<percent> ::= percent in NR1 format
:POWer:SWITCh:RDS <value>[suffix] (see page 933)	:POWer:SWITCh:RDS? (see page 933)	<value> ::= Rds(on) value in NR3 format [suffix] ::= {OHM mOHM}
:POWer:SWITCh:VCE <value>[suffix] (see page 934)	:POWer:SWITCh:VCE? (see page 934)	<value> ::= Vce(sat) value in NR3 format [suffix] ::= {V mV}
:POWer:SWITCh:VREFere nce <percent> (see page 935)	:POWer:SWITCh:VREFere nce? (see page 935)	<percent> ::= percent in NR1 format
:POWer:TRANsient:APPL y (see page 936)	n/a	n/a
:POWer:TRANsient:EXIT (see page 937)	n/a	n/a
:POWer:TRANsient:IINi tial <value>[suffix] (see page 938)	:POWer:TRANsient:IINi tial? (see page 938)	<value> ::= Initial current value in NR3 format [suffix] ::= {A mA}
:POWer:TRANsient:INEW <value>[suffix] (see page 939)	:POWer:TRANsient:INEW ? (see page 939)	<value> ::= New current value in NR3 format [suffix] ::= {A mA}
:POWer:TRANsient:NEXT (see page 940)	n/a	n/a

:POWer:CLResponse?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse?

The :POWer:CLResponse? query returns the Control Loop Response (Bode) power analysis settings.

Return Format <settings_string><NL>

For example, the query returns the following string when issued after the *RST command.

```
:POW:CLR:SOUR:INP CHAN1;OUTP CHAN2;:POW:CLR:FREQ:STAR +100E+00;
STOP +20.000000E+06;:POW:CLR:WGEN:VOLT +200.0E-03;LOAD FIFT
```

- See Also**
- [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:APPLy

N (see [page 1708](#))

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 and/or phase as a function of frequency.

You can use the :POWer:CLResponse:TRACe command to specify whether to include gain, phase, both gain and phase, or neither in the frequency response analysis results.

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**
- ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:SINGle"](#) on page 853
 - [":POWer:CLResponse:FREQuency:STARt"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:TRACe"](#) on page 859
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:DATA?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse:DATA? [SWEep | SINGle]

The :POWer:CLResponse:DATA? query returns data from the Control Loop Response (Bode) power analysis.

The comma-separated value format is suitable for spreadsheet analysis.

You can use the :POWer:CLResponse:TRACe command to specify whether to include gain, phase, both gain and phase, or neither in the frequency response analysis results.

The SWEep or SINGle option specifies whether to get the data from a sweep or single-frequency analysis (see :POWer:CLResponse:FREQuency:MODE). If this option is not specified, the data from the sweep analysis is returned by default.

Return Format <binary_block><NL>

<binary_block> ::= comma-separated data with newlines at the end of each row

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLy"](#) on page 846
 - [":POWer:CLResponse:DATA:GMARgin?"](#) on page 848
 - [":POWer:CLResponse:DATA:GMARgin:FREQuency?"](#) on page 849
 - [":POWer:CLResponse:DATA:PMARgin?"](#) on page 850
 - [":POWer:CLResponse:DATA:PMARgin:FREQuency?"](#) on page 851
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:SINGle"](#) on page 853
 - [":POWer:CLResponse:FREQuency:STARt"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:TRACe"](#) on page 859
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:DATA:GMARgin?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse:DATA:GMARgin?

After the Control Loop Response (Bode) power analysis has been performed (see :POWer:CLResponse:APPLY), the :POWer:CLResponse:DATA:GMARgin? query returns the gain margin in dB.

Return Format <gain_margin><NL>

<gain_margin> ::= gain margin in dB in NR3 format.

The query returns +9.9E+37 if the value cannot be calculated from the last sweep (that is, if there are no crossover points).

- See Also**
- [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:DATA:GMARgin:FREQuency?"](#) on page 849
 - [":POWer:CLResponse:DATA:PMARgin?"](#) on page 850
 - [":POWer:CLResponse:DATA:PMARgin:FREQuency?"](#) on page 851

:POWer:CLResponse:DATA:GMARgin:FREQuency?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse:DATA:GMARgin:FREQuency?

After the Control Loop Response (Bode) power analysis has been performed (see :POWer:CLResponse:APPLy), the :POWer:CLResponse:DATA:GMARgin:FREQuency? query returns the 0° phase crossover frequency in Hz.

Return Format <frequency><NL>

<frequency> ::= 0 degrees phase crossover frequency in Hz in NR3 format

The query returns +9.9E+37 if the value cannot be calculated from the last sweep (that is, if there are no crossover points).

- See Also**
- [":POWer:CLResponse:APPLy"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:DATA:GMARgin?"](#) on page 848
 - [":POWer:CLResponse:DATA:PMARgin?"](#) on page 850
 - [":POWer:CLResponse:DATA:PMARgin:FREQuency?"](#) on page 851

:POWer:CLResponse:DATA:PMARgin?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse:DATA:PMARgin?

After the Control Loop Response (Bode) power analysis has been performed (see :POWer:CLResponse:APPLY), the :POWer:CLResponse:DATA:PMARgin? query returns the phase margin in degrees.

Return Format <phase_margin><NL>

<phase_margin> ::= phase margin in degrees in NR3 format.

The query returns +9.9E+37 if the value cannot be calculated from the last sweep (that is, if there are no crossover points).

- See Also**
- [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:DATA:GMARgin?"](#) on page 848
 - [":POWer:CLResponse:DATA:GMARgin:FREQuency?"](#) on page 849
 - [":POWer:CLResponse:DATA:PMARgin:FREQuency?"](#) on page 851

:POWer:CLResponse:DATA:PMARgin:FREQuency?

N (see [page 1708](#))

Query Syntax :POWer:CLResponse:DATA:PMARgin:FREQuency?

After the Control Loop Response (Bode) power analysis has been performed (see :POWer:CLResponse:APPLy), the :POWer:CLResponse:DATA:PMARgin:FREQuency? query returns the 0 dB gain crossover frequency in Hz.

Return Format <frequency><NL>

<frequency> ::= 0dB gain crossover frequency in Hz in NR3 format.

The query returns +9.9E+37 if the value cannot be calculated from the last sweep (that is, if there are no crossover points).

- See Also**
- [":POWer:CLResponse:APPLy"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:DATA:GMARgin?"](#) on page 848
 - [":POWer:CLResponse:DATA:GMARgin:FREQuency?"](#) on page 849
 - [":POWer:CLResponse:DATA:PMARgin?"](#) on page 850

:POWer:CLResponse:FREQuency:MODE

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:FREQuency:MODE <mode>

<mode> ::= {SWEep | SINGle}

The :POWer:CLResponse:FREQuency:MODE command specifies whether the analysis should be performed by sweeping through a range of frequencies (SWEep) or at a single frequency (SINGle).

The SINGle mode is useful for evaluating amplitudes at a single frequency, for example, near the expected 0 dB cross-over frequency. After running the test at a single frequency, you can manually adjust (increase) the waveform generator's amplitude until you begin to observe distortion in the waveforms on the oscilloscope's display. You can then use that amplitude at all frequencies in SWEep mode, or you can evaluate amplitudes at other frequencies in order to determine an optimized amplitude profile (see :POWer:CLResponse:WGEN:VOLTage:PROFile).

Query Syntax :POWer:CLResponse:FREQuency:MODE?

The :POWer:CLResponse:FREQuency:MODE? query returns the frequency mode setting.

Return Format <mode><NL>

<mode> ::= {SWE | SING}

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:SINGle"](#) on page 853
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:FREQuency:SINGle

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:FREQuency:SINGle <value>[suffix]
 <value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
 | 20000000}
 [suffix] ::= {Hz | kHz | MHz}

The :POWer:CLResponse:FREQuency:SINGle command sets the single frequency 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:SINGle?

The :POWer:CLResponse:FREQuency:SINGle? query returns the single frequency setting.

Return Format <value><NL>
 <value> ::= {20 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
 | 20000000}

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLy"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:FREQuency:START

N (see [page 1708](#))

- 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?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:FREQuency:STOP

N (see [page 1708](#))

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?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:PPDecade

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:PPDecade <pts>

<pts> ::= {10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100}

The :POWer:CLResponse:PPDecade command selects the number of frequency test points per decade (in the log scale).

Query Syntax :POWer:CLResponse:PPDecade?

The :POWer:CLResponse:PPDecade? query returns the points per decade setting.

Return Format <pts><NL>

<pts> ::= {10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100}

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:SOURce:INPut

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:SOURce:INPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :POWer:CLResponse:SOURce:INPut command selects the oscilloscope channel that is probing the power supply input.

Query Syntax :POWer:CLResponse:SOURce:INPut?

The :POWer:CLResponse:SOURce:INPut? query returns the channel selection.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLy"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:STARt"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:SOURce:OUTPut

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:SOURce:OUTPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :POWer:CLResponse:SOURce:OUTPut command selects the oscilloscope channel that is probing the power supply output.

Query Syntax :POWer:CLResponse:SOURce:OUTPut?

The :POWer:CLResponse:SOURce:OUTPut? query returns the channel selection.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:TRACe

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:TRACe <selection>

<selection> ::= {NONE | ALL | GAIN | PHASe}[, {GAIN | PHASe}]

The :POWer:CLResponse:TRACe command specifies whether to include gain, phase, both gain and phase, or neither in the control loop response analysis results.

NOTE

This command affects the oscilloscope's front panel graphical user interface (plot and table) as well as when saving analysis data.

Query Syntax :POWer:CLResponse:TRACe?

The :POWer:CLResponse:TRACe? query returns a comma-separated list of the types of data that are currently included in the control loop response analysis results, or "NONE" if neither gain nor phase data is included.

Return Format <selection_list><NL>

<selection_list> ::= {"NONE" | "GAIN" | "PHASe" | "GAIN, PHASe"}

- See Also**
- [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847

:POWer:CLResponse:WGEN:LOAD

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:WGEN:LOAD <impedance>

<impedance> ::= {ONEMeg | FIFTy}

The :POWer:CLResponse:WGEN:LOAD command sets the waveform generator expected output load impedance.

The output impedance of the Gen Out signal 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 :POWer:CLResponse:WGEN:LOAD?

The :POWer:CLResponse:WGEN:LOAD? query returns the waveform generator expected output load impedance setting.

Return Format <impedance><NL>

<impedance> ::= {ONEM | FIFT}

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:WGEN:VOLTage

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:WGEN:VOLTage <amplitude>[,<range>]

<amplitude> ::= amplitude in volts in NR3 format

<range> ::= {F10MILLIHZ | F100MILLIHZ | F1HZ | {F10HZ | F20HZ} | F100HZ
| F1KHZ | F10KHZ | F100KHZ | F1MHZ | F10MHZ | {F20MHZ | F100MHZ}}

The :POWer:CLResponse:WGEN:VOLTage command sets the waveform generator output amplitude(s).

When the waveform generator amplitude profile is enabled (with the :POWer:CLResponse:WGEN:VOLTage:PROFile command), you can set an initial ramp amplitude for each frequency range.

Without the <range> parameter, this command sets the waveform generator output amplitude used when the amplitude profile is disabled.

NOTE

The F20HZ and F20MHZ <range> options are allowed for backward compatibility; however, they map to the F10HZ and F100MHZ boundaries, respectively.

Query Syntax :POWer:CLResponse:WGEN:VOLTage? [<range>]

The :POWer:CLResponse:WGEN:VOLTage? query returns the waveform generator output amplitude setting(s).

Return Format <amplitude><NL>

<amplitude> ::= amplitude in volts in NR3 format

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage:PROFile"](#) on page 862

:POWer:CLResponse:WGEN:VOLTage:PROFile

N (see [page 1708](#))

Command Syntax :POWer:CLResponse:WGEN:VOLTage:PROFile {{0 | OFF} | {1 | ON}}

The :POWer:CLResponse:WGEN:VOLTage:PROFile command enables or disables the ability to set initial waveform generator ramp amplitudes for each frequency range.

With amplitude profiling, you can use lower amplitudes at frequencies where the device under test (DUT) is sensitive to distortion and use higher amplitudes where the DUT is less sensitive to distortion. Power supply feedback networks are typically most sensitive near the 0 dB cross-over frequency.

You can often observe distortions during the test. If the input test sine wave begins to look lopsided, clipped, or somewhat triangular in shape (nonsinusoidal), you are probably encountering distortion due to overdriving your DUT. Optimizing test amplitudes to achieve the best dynamic range measurements is often an iterative process of running your frequency response measurements multiple times.

Query Syntax :POWer:CLResponse:WGEN:VOLTage:PROFile?

The :POWer:CLResponse:WGEN:VOLTage:PROFile? query returns the voltage profile setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":POWer:CLResponse?"](#) on page 845
 - [":POWer:CLResponse:APPLY"](#) on page 846
 - [":POWer:CLResponse:DATA?"](#) on page 847
 - [":POWer:CLResponse:FREQuency:MODE"](#) on page 852
 - [":POWer:CLResponse:FREQuency:START"](#) on page 854
 - [":POWer:CLResponse:FREQuency:STOP"](#) on page 855
 - [":POWer:CLResponse:PPDecade"](#) on page 856
 - [":POWer:CLResponse:SOURce:INPut"](#) on page 857
 - [":POWer:CLResponse:SOURce:OUTPut"](#) on page 858
 - [":POWer:CLResponse:WGEN:LOAD"](#) on page 860
 - [":POWer:CLResponse:WGEN:VOLTage"](#) on page 861

:POWer:DESKew

N (see [page 1708](#))

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 *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 927
 - **" :POWer:SIGNals:SOURce:CURREnt<i>"** on page 926

:POWer:EFFiciency:APPLy

N (see [page 1708](#))

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 865
 - [":MEASure:EFFiciency"](#) on page 778
 - [":MEASure:IPOWer"](#) on page 781
 - [":MEASure:OPOWer"](#) on page 784

:POWer:EFFiciency:TYPE

N (see [page 1708](#))

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 864
 - [":MEASure:EFFiciency"](#) on page 778
 - [":MEASure:IPOWer"](#) on page 781
 - [":MEASure:OPOWer"](#) on page 784

:POWer:ENABle

N (see [page 1708](#))

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 29](#), “:MEASure Power Commands,” starting on page 769

:POWer:HARMonics:APPLy

N (see [page 1708](#))

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 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:DATA?

N (see [page 1708](#))

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 867
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:DISPlay

N (see [page 1708](#))

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 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:FAILcount?

N (see [page 1708](#))

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 875
 - [":POWer:HARMonics:APPLy"](#) on page 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:LINE

N (see [page 1708](#))

Command Syntax :POWer:HARMonics:LINE <frequency>

<frequency> ::= {F50 | F60 | F400 | AUTO}

The :POWer:HARMonics:LINE command specifies the line frequency setting for the current harmonics analysis:

- F50 – 50 Hz.
- F60 – 60 Hz.
- F400 – 400 Hz.
- AUTO – Automatically determines the frequency of the Current waveform.

Query Syntax :POWer:HARMonics:LINE?

The :POWer:HARMonics:LINE query returns the line frequency setting.

Return Format <frequency><NL>

<frequency> ::= {F50 | F60 | F400 | AUTO}

- See Also**
- [":POWer:HARMonics:APPLY"](#) on page 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:POWerfactor?

N (see [page 1708](#))

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 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:RPOWer

N (see [page 1708](#))

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 876
 - [":POWer:HARMonics:RPOWer:USER"](#) on page 874

:POWer:HARMonics:RPOWer:USER

N (see [page 1708](#))

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 876
 - [":POWer:HARMonics:RPOWer"](#) on page 873

:POWer:HARMonics:RUNCount?

N (see [page 1708](#))

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 870
 - [":POWer:HARMonics:APPLy"](#) on page 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:STANdard

N (see [page 1708](#))

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 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STATus?"](#) on page 877
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:STATus?

N (see [page 1708](#))

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 875
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:APPLY"](#) on page 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:THD?"](#) on page 878

:POWer:HARMonics:THD?

N (see [page 1708](#))

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 867
 - [":POWer:HARMonics:DATA?"](#) on page 868
 - [":POWer:HARMonics:DISPlay"](#) on page 869
 - [":POWer:HARMonics:FAILcount?"](#) on page 870
 - [":POWer:HARMonics:LINE"](#) on page 871
 - [":POWer:HARMonics:POWerfactor?"](#) on page 872
 - [":POWer:HARMonics:RUNCount?"](#) on page 875
 - [":POWer:HARMonics:STANdard"](#) on page 876
 - [":POWer:HARMonics:STATus?"](#) on page 877

:POWer:INRush:APPLy

N (see [page 1708](#))

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**
- [":POWer:ITYPE"](#) on page 882
 - [":POWer:INRush:EXIT"](#) on page 880
 - [":POWer:INRush:NEXT"](#) on page 881
 - [":MEASure:PCURrent"](#) on page 785

:POWer:INRush:EXIT

N (see [page 1708](#))

Command Syntax :POWer:INRush:EXIT

The :POWer:INRush:EXIT command exits (stops) the inrush current power analysis.

This command is equivalent to selecting **Exit** in the user interface during the analysis.

- See Also**
- [":POWer:INRush:APPLY"](#) on page 879
 - [":POWer:INRush:NEXT"](#) on page 881
 - [":POWer:ITYPE"](#) on page 882

:POWer:INRush:NEXT

N (see [page 1708](#))

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 selecting **Next** in the user interface when prompted during the analysis.

- See Also**
- [":POWer:INRush:APPLY"](#) on page 879
 - [":POWer:INRush:EXIT"](#) on page 880
 - [":POWer:ITYPE"](#) on page 882

:POWer:ITYPE

N (see [page 1708](#))

Command Syntax :POWer:ITYPE <type>

<type> ::= {DC | AC}

The :POWer:ITYPE command specifies the type of power that is being converted from the input (DC or AC). Your selection affects how the measurements are made.

This setting is used in the Inrush Current and Turn On/Turn Off tests.

Query Syntax :POWer:ITYPE?

The :POWer:ITYPE? query returns the input power type setting.

Return Format <type><NL>

<type> ::= {DC | AC}

- See Also**
- [":POWer:INRush:APPLY"](#) on page 879
 - [":POWer:ONOff:APPLY"](#) on page 887

:POWer:MODulation:APPLy

N (see [page 1708](#))

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 884
 - [":POWer:MODulation:TYPE"](#) on page 885
 - [":MEASure:VAverage"](#) on page 757
 - [":MEASure:VRMS"](#) on page 763
 - [":MEASure:VRATio"](#) on page 762
 - [":MEASure:PERiod"](#) on page 718
 - [":MEASure:FREQuency"](#) on page 693
 - [":MEASure:PWIDth"](#) on page 722
 - [":MEASure:NWIDth"](#) on page 714
 - [":MEASure:DUTYcycle"](#) on page 687
 - [":MEASure:RISetime"](#) on page 728
 - [":MEASure:FALLtime"](#) on page 688

:POWer:MODulation:SOURce

N (see [page 1708](#))

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 883
 - [":POWer:MODulation:TYPE"](#) on page 885

:POWer:MODulation:TYPE

N (see [page 1708](#))

Command Syntax :POWer:MODulation:TYPE <modulation>

```
<modulation> ::= {VAVerage | ACRMs | VRATio | PERiod | FREQuency  
| PWIDith | NWIDth | DUTYcycle | NDUTYcycle | RISetime | 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
- NDUTYcycle
- RISetime
- FALLtime

Query Syntax :POWer:MODulation:TYPE?

The :POWer:MODulation:TYPE query returns the modulation type setting.

Return Format <modulation><NL>

```
<modulation> ::= {VAV | ACRM | VRAT | PER | FREQ | PWID | NWID | DUTY  
| NDUT | RIS | FALL}
```

- See Also**
- [":POWer:MODulation:SOURce"](#) on page 884
 - [":POWer:MODulation:APPLY"](#) on page 883
 - [":MEASure:VAVerage"](#) on page 757
 - [":MEASure:VRMS"](#) on page 763
 - [":MEASure:VRATio"](#) on page 762
 - [":MEASure:PERiod"](#) on page 718
 - [":MEASure:FREQuency"](#) on page 693
 - [":MEASure:PWIDth"](#) on page 722
 - [":MEASure:NWIDth"](#) on page 714
 - [":MEASure:DUTYcycle"](#) on page 687
 - [":MEASure:RISetime"](#) on page 728

- **":MEASure:FALLtime"** on page 688

:POWer:ONOFF:APPLy

N (see [page 1708](#))

Command Syntax :POWer:ONOFF:APPLy

The :POWer:ONOFF:APPLy command applies the selected turn on/off analysis test (:POWer:ONOFF:TEST).

- See Also**
- [":POWer:SIGNals:VSTeady:ONOFF:OFF"](#) on page 923
 - [":POWer:SIGNals:VSTeady:ONOFF:ON"](#) on page 924
 - [":POWer:ITYPE"](#) on page 882
 - [":POWer:ONOFF:THResholds"](#) on page 891
 - [":POWer:ONOFF:TEST"](#) on page 890
 - [":POWer:ONOFF:EXIT"](#) on page 888
 - [":POWer:ONOFF:NEXT"](#) on page 889
 - [":MEASure:ONTime"](#) on page 783
 - [":MEASure:OFFTime"](#) on page 782

:POWer:ONOFF:EXIT

N (see [page 1708](#))

Command Syntax :POWer:ONOFF:EXIT

The :POWer:ONOFF:EXIT command exits (stops) the turn on time/turn off time analysis.

This command is equivalent to selecting **Exit** in the user interface during the analysis.

- See Also**
- [":POWer:ONOFF:THResholds"](#) on page 891
 - [":POWer:ITYPE"](#) on page 882
 - [":POWer:ONOFF:APPLY"](#) on page 887
 - [":POWer:ONOFF:NEXT"](#) on page 889
 - [":POWer:ONOFF:TEST"](#) on page 890

:POWer:ONOff:NEXT

N (see [page 1708](#))

Command Syntax :POWer:ONOff:NEXT

The :POWer:ONOff:NEXT command goes to the next step of the turn on/turn off analysis.

This command is equivalent to selecting **Next** in the user interface when prompted during the analysis.

- See Also**
- [":POWer:ONOff:THResholds"](#) on page 891
 - [":POWer:ITYPE"](#) on page 882
 - [":POWer:ONOff:APPLY"](#) on page 887
 - [":POWer:ONOff:EXIT"](#) on page 888
 - [":POWer:ONOff:TEST"](#) on page 890

:POWer:ONOFF:TEST

N (see [page 1708](#))

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:THResholds"](#) on page 891
 - [":POWer:ITYPE"](#) on page 882
 - [":POWer:ONOFF:APPLY"](#) on page 887
 - [":POWer:ONOFF:EXIT"](#) on page 888
 - [":POWer:ONOFF:NEXT"](#) on page 889

:POWer:ONOff:THResholds

N (see [page 1708](#))

Command Syntax :POWer:ONOff:THResholds <type>,<input_thr>,<output_thr>

<type> ::= {ON | OFF}

<input_thr> ::= percent from 0-100 in NR1 format

<output_thr> ::= percent from 0-100 in NR1 format

The :POWer:ONOff:THResholds command specifies the input and output thresholds used in the Turn On/Turn Off analysis.

Turn On analysis determines how fast a turned on power supply takes to reach some percent of its steady state output. Turn on time is the time between T2 and T1 where:

- T1 = when the input voltage first rises to some percent (typically the 10% threshold) of its maximum amplitude.
- T2 = when the output DC voltage rises to some percent (typically the 90% threshold) of its maximum amplitude.

Turn Off analysis determines how fast a turned off power supply takes to reduce its output voltage to some percent of maximum. Turn off time is the time between T2 and T1 where:

- T1 = when the input voltage last falls to some percent (typically the 10% threshold) of its maximum amplitude.
- T2 = when the output DC voltage last falls to some percent (typically the 10% threshold) of its maximum amplitude.

Query Syntax :POWer:ONOff:THResholds? <type>

The :POWer:ONOff:THResholds? query returns the input and output threshold settings for the turn on/turn off analysis type.

Return Format <input_thr>,<output_thr><NL>

<input_thr> ::= percent from 0-100 in NR1 format

<output_thr> ::= percent from 0-100 in NR1 format

- See Also**
- [":POWer:SIGNALs:VSTeady:ONOff:OFF"](#) on page 923
 - [":POWer:SIGNALs:VSTeady:ONOff:ON"](#) on page 924
 - [":POWer:ITYPE"](#) on page 882
 - [":POWer:ONOff:APPLY"](#) on page 887
 - [":POWer:ONOff:TEST"](#) on page 890
 - [":POWer:ONOff:EXIT"](#) on page 888
 - [":POWer:ONOff:NEXT"](#) on page 889

- **":MEASure:ONTime"** on page 783
- **":MEASure:OFFTime"** on page 782

:POWer:PSRR?

N (see [page 1708](#))

Query Syntax :POWer:PSRR?

The :POWer:PSRR? query returns the Power Supply Rejection Ratio (PSRR) power analysis settings.

Return Format <settings_string><NL>

For example, the query returns the following string when issued after the *RST command.

```
:POW:PSRR:SOUR:INP CHAN1;OUTP CHAN2;:POW:PSRR:FREQ:STAR +100E+00;
STOP +20.000000E+06;:POW:PSRR:WGEN:VOLT +200.0E-03;LOAD FIFT
```

- See Also**
- [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:APPLy

N (see [page 1708](#))

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.

You can use the :POWer:PSRR:TRACe command to specify whether to include gain data in the PSRR analysis results.

- See Also
- ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:FREQuency:SINGLE"](#) on page 899
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:TRACe"](#) on page 903
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:DATA?

N (see [page 1708](#))

Query Syntax :POWer:PSRR:DATA? [SWEep | SINGLE]

The :POWer:PSRR:DATA? query returns data from the Power Supply Rejection Ratio (PSRR) power analysis.

The comma-separated value format is suitable for spreadsheet analysis.

You can use the :POWer:PSRR:TRACe command to specify whether to include gain data in the PSRR analysis results.

The SWEep or SINGLE option specifies whether to get the data from a sweep or single-frequency analysis (see :POWer:PSRR:FREQuency:MODE). If this option is not specified, the data from the sweep analysis is returned by default.

Return Format <binary_block><NL>

<binary_block> ::= comma-separated data with newlines at the end of each row

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:FREQuency:SINGLE"](#) on page 899
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:TRACe"](#) on page 903
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:FREQuency:MAXimum

N (see [page 1708](#))

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?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:FREQuency:MINimum

N (see [page 1708](#))

- 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?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:FREQuency:MODE

N (see [page 1708](#))

Command Syntax :POWer:PSRR:FREQuency:MODE <mode>

<mode> ::= {SWEep | SINGle}

The :POWer:PSRR:FREQuency:MODE command specifies whether the analysis should be performed by sweeping through a range of frequencies (SWEep) or at a single frequency (SINGle).

The SINGle mode is useful for evaluating amplitudes at a single frequency. After running the test at a single frequency, you can manually adjust (increase) the waveform generator's amplitude until you begin to observe distortion in the waveforms on the oscilloscope's display. You can then use that amplitude at all frequencies in SWEep mode, or you can evaluate amplitudes at other frequencies in order to determine an optimized amplitude profile (see :POWer:PSRR:WGEN:VOLTage:PROFile).

Query Syntax :POWer:PSRR:FREQuency:MODE?

The :POWer:PSRR:FREQuency:MODE? query returns the frequency mode setting.

Return Format <mode><NL>

<mode> ::= {SWE | SING}

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:SINGle"](#) on page 899
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:FREQuency:SINGle

N (see [page 1708](#))

Command Syntax :POWer:PSRR:FREQuency:SINGle <value>[suffix]

<value> ::= {1 | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
| 20000000}

[suffix] ::= {Hz | kHz | MHz}

The :POWer:PSRR:FREQuency:SINGle command sets the single frequency value. The measurement is displayed on a log scale, so you can select from decade values.

Query Syntax :POWer:PSRR:FREQuency:SINGle?

The :POWer:PSRR:FREQuency:SINGle query returns the single frequency setting.

Return Format <value><NL>

<value> ::= {1 | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000
| 20000000}

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:PPDecade

N (see [page 1708](#))

Command Syntax :POWer:PSRR:PPDecade <pts>

<pts> ::= {10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100}

The :POWer:PSRR:PPDecade command selects the number of frequency test points per decade (in the log scale).

Query Syntax :POWer:PSRR:PPDecade?

The :POWer:PSRR:PPDecade? query returns the points per decade setting.

Return Format <pts><NL>

<pts> ::= {10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100}

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:SOURce:INPut

N (see [page 1708](#))

Command Syntax :POWer:PSRR:SOURce:INPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :POWer:PSRR:SOURce:INPut command selects the oscilloscope channel that is probing the power supply input.

Query Syntax :POWer:PSRR:SOURce:INPut?

The :POWer:PSRR:SOURce:INPut? query returns the channel selection.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:SOURce:OUTPut

N (see [page 1708](#))

Command Syntax :POWer:PSRR:SOURce:OUTPut <source>

<source> ::= CHANnel<n>

<n> ::= 1 to (# analog channels) in NR1 format

The :POWer:PSRR:SOURce:OUTPut command selects the oscilloscope channel that is probing the power supply output.

Query Syntax :POWer:PSRR:SOURce:OUTPut?

The :POWer:PSRR:SOURce:OUTPut? query returns the channel selection.

Return Format <source><NL>

<source> ::= CHAN<n>

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:TRACe

N (see [page 1708](#))

Command Syntax :POWer:PSRR:TRACe <selection>

<selection> ::= {NONE | GAIN}

The :POWer:PSRR:TRACe command specifies whether to include gain (PSRR) data in the power supply rejection ratio analysis results.

NOTE

This command affects the oscilloscope's front panel graphical user interface (plot and table) as well as when saving analysis data.

Query Syntax :POWer:PSRR:TRACe?

The :POWer:PSRR:TRACe? query returns the type of data that is currently included in the PSRR analysis results, or "NONE" if the gain data is not included.

Return Format <selection_list><NL>

<selection_list> ::= {"NONE" | "GAIN"}

- See Also**
- [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895

:POWer:PSRR:WGEN:LOAD

N (see [page 1708](#))

Command Syntax :POWer:PSRR:WGEN:LOAD <impedance>

<impedance> ::= {ONEMeg | FIFTy}

The :POWer:PSRR:WGEN:LOAD command sets the waveform generator expected output load impedance.

The output impedance of the Gen Out signal 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 :POWer:PSRR:WGEN:LOAD?

The :POWer:PSRR:WGEN:LOAD? query returns the waveform generator expected output load impedance setting.

Return Format <impedance><NL>

<impedance> ::= {ONEM | FIFT}

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:WGEN:VOLTage

N (see [page 1708](#))

Command Syntax :POWer:PSRR:WGEN:VOLTage <amplitude>[,<range>]

<amplitude> ::= amplitude in volts in NR3 format

<range> ::= {F10MILLIHZ | F100MILLIHZ | F1HZ | {F10HZ | F20HZ} | F100HZ
| F1KHZ | F10KHZ | F100KHZ | F1MHZ | F10MHZ | {F20MHZ | F100MHZ}}

The :POWer:PSRR:WGEN:VOLTage command sets the waveform generator output amplitude(s).

When the waveform generator amplitude profile is enabled (with the :POWer:PSRR:WGEN:VOLTage:PROFile command), you can set an initial ramp amplitude for each frequency range.

Without the <range> parameter, this command sets the waveform generator output amplitude used when the amplitude profile is disabled.

NOTE

The F20HZ and F20MHZ <range> options are allowed for backward compatibility; however, they map to the F10HZ and F100MHZ boundaries, respectively.

Query Syntax :POWer:PSRR:WGEN:VOLTage? [<range>]

The :POWer:PSRR:WGEN:VOLTage? query returns the waveform generator output amplitude setting(s).

Return Format <amplitude><NL>

<amplitude> ::= amplitude in volts in NR3 format

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLy"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage:PROFile"](#) on page 906

:POWer:PSRR:WGEN:VOLTage:PROFile

N (see [page 1708](#))

Command Syntax :POWer:PSRR:WGEN:VOLTage:PROFile {{0 | OFF} | {1 | ON}}

The :POWer:PSRR:WGEN:VOLTage:PROFile command enables or disables the ability to set initial waveform generator ramp amplitudes for each frequency range.

With amplitude profiling, you can use lower amplitudes at frequencies where the device under test (DUT) is sensitive to distortion and use higher amplitudes where the DUT is less sensitive to distortion.

You can often observe distortions during the test. If the input test sine wave begins to look lopsided, clipped, or somewhat triangular in shape (nonsinusoidal), you are probably encountering distortion due to overdriving your DUT. Optimizing test amplitudes to achieve the best dynamic range measurements is often an iterative process of running your frequency response measurements multiple times.

Query Syntax :POWer:PSRR:WGEN:VOLTage:PROFile?

The :POWer:PSRR:WGEN:VOLTage:PROFile? query returns the voltage profile setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":POWer:PSRR?"](#) on page 893
 - [":POWer:PSRR:APPLY"](#) on page 894
 - [":POWer:PSRR:DATA?"](#) on page 895
 - [":POWer:PSRR:FREQuency:MAXimum"](#) on page 896
 - [":POWer:PSRR:FREQuency:MINimum"](#) on page 897
 - [":POWer:PSRR:FREQuency:MODE"](#) on page 898
 - [":POWer:PSRR:PPDecade"](#) on page 900
 - [":POWer:PSRR:SOURce:INPut"](#) on page 901
 - [":POWer:PSRR:SOURce:OUTPut"](#) on page 902
 - [":POWer:PSRR:WGEN:LOAD"](#) on page 904
 - [":POWer:PSRR:WGEN:VOLTage"](#) on page 905

:POWer:QUALity:APPLy

N (see [page 1708](#))

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 780
 - [":MEASure:REAL"](#) on page 789
 - [":MEASure:APParent"](#) on page 775
 - [":MEASure:REACTive"](#) on page 788
 - [":MEASure:CRESt"](#) on page 777
 - [":MEASure:ANGLE"](#) on page 774

:POWer:RIPPlE:APPLy

N (see [page 1708](#))

Command Syntax :POWer:RIPPlE:APPLy

The :POWer:RIPPlE:APPLy command applies the output ripple analysis.

See Also • [":MEASure:RIPPlE"](#) on page 790

:POWer:SIGNals:AUTosetup

N (see [page 1708](#))

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 869
 - [":POWer:EFFiciency:APPLy"](#) on page 864
 - [":POWer:RIPPlE:APPLy"](#) on page 908
 - [":POWer:MODulation:APPLy"](#) on page 883
 - [":POWer:QUALity:APPLy"](#) on page 907
 - [":POWer:SLEW:APPLy"](#) on page 928
 - [":POWer:SWITCh:APPLy"](#) on page 930
 - [":POWer:SIGNals:CYCLes:HARMonics"](#) on page 910
 - [":POWer:SIGNals:CYCLes:QUALity"](#) on page 911
 - [":POWer:SIGNals:DURation:EFFiciency"](#) on page 912
 - [":POWer:SIGNals:DURation:MODulation"](#) on page 913
 - [":POWer:SIGNals:DURation:RIPPlE"](#) on page 916
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURRent<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:CYCLes:HARMonics

N (see [page 1708](#))

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 869
 - [":POWer:HARMonics:APPLy"](#) on page 867
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURRent<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:CYCLes:QUALity

N (see [page 1708](#))

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 907
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:EFFiciency

N (see [page 1708](#))

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 864
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:MODulation

N (see [page 1708](#))

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 883
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:ONOFF:OFF

N (see [page 1708](#))

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 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:ONOFF:OFF"](#) on page 921
 - [":POWer:SIGNals:VSTeady:ONOFF:OFF"](#) on page 923
 - [":POWer:SIGNals:SOURce:CURRent<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:ONOff:ON

N (see [page 1708](#))

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 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:ONOff:ON"](#) on page 922
 - [":POWer:SIGNals:VSTeady:ONOff:ON"](#) on page 924
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:RIPple

N (see [page 1708](#))

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 908
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURRent<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:DURation:TRANsient

N (see [page 1708](#))

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 936
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VSTeady:TRANsient"](#) on page 925
 - [":POWer:SIGNals:SOURce:CURRENT<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:IEXPeCted

N (see [page 1708](#))

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 879
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:INRush"](#) on page 920
 - [":POWer:SIGNals:SOURce:CURRent<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:OVERshoot

N (see [page 1708](#))

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 936
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:TRANsient"](#) on page 917
 - [":POWer:SIGNals:IEXPeCted"](#) on page 918
 - [":POWer:SIGNals:VSTeady:TRANsient"](#) on page 925
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VMAXimum:INRush

N (see [page 1708](#))

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.

When the :POWer:ITYPE is DC, this command defines the maximum DC input voltage amplitude value. The values can be negative.

When the :POWer:ITYPE is AC, this command defines the maximum peak-to-peak input voltage. Only positive values are allowed.

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:ITYPE"](#) on page 882
 - [":POWer:INRush:APPLY"](#) on page 879
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURRENT<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VMAXimum:ONOff:OFF

N (see [page 1708](#))

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.

When the :POWer:ITYPE is DC, this command defines the maximum DC input voltage amplitude value. The values can be negative.

When the :POWer:ITYPE is AC, this command defines the maximum peak-to-peak input voltage. Only positive values are allowed.

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:ITYPE"](#) on page 882
 - [":POWer:ONOff:APPLY"](#) on page 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:ONOff:OFF"](#) on page 914
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VSTeady:ONOff:OFF"](#) on page 923
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VMAXimum:ONOff:ON

N (see [page 1708](#))

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.

When the :POWer:ITYPE is DC, this command defines the maximum DC input voltage amplitude value. The values can be negative.

When the :POWer:ITYPE is AC, this command defines the maximum peak-to-peak input voltage. Only positive values are allowed.

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:ITYPE"](#) on page 882
 - [":POWer:ONOff:APPLY"](#) on page 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:ONOff:ON"](#) on page 915
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VSTeady:ONOff:ON"](#) on page 924
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VSTeady:ONOff:OFF

N (see [page 1708](#))

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 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:ONOff:OFF"](#) on page 914
 - [":POWer:SIGNals:IEXPected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:ONOff:OFF"](#) on page 921
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VSTeady:ONOff:ON

N (see [page 1708](#))

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 887
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:ONOff:ON"](#) on page 915
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:ONOff:ON"](#) on page 922
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:VSTeady:TRANsient

N (see [page 1708](#))

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 936
 - [":POWer:SIGNals:AUTosetup"](#) on page 909
 - [":POWer:SIGNals:DURation:TRANsient"](#) on page 917
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:SOURce:CURRent<i>

N (see [page 1708](#))

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 909
 - [":POWer:SIGNals:CYCLes:HARMonics"](#) on page 910
 - [":POWer:SIGNals:CYCLes:QUALity"](#) on page 911
 - [":POWer:SIGNals:DURation:EFFiciency"](#) on page 912
 - [":POWer:SIGNals:DURation:MODulation"](#) on page 913
 - [":POWer:SIGNals:DURation:ONOff:OFF"](#) on page 914
 - [":POWer:SIGNals:DURation:ONOff:ON"](#) on page 915
 - [":POWer:SIGNals:DURation:RIPple"](#) on page 916
 - [":POWer:SIGNals:DURation:TRANsient"](#) on page 917
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:INRush"](#) on page 920
 - [":POWer:SIGNals:VMAXimum:ONOff:OFF"](#) on page 921
 - [":POWer:SIGNals:VMAXimum:ONOff:ON"](#) on page 922
 - [":POWer:SIGNals:VSTeady:ONOff:OFF"](#) on page 923
 - [":POWer:SIGNals:VSTeady:ONOff:ON"](#) on page 924
 - [":POWer:SIGNals:VSTeady:TRANsient"](#) on page 925
 - [":POWer:SIGNals:SOURce:VOLTage<i>"](#) on page 927

:POWer:SIGNals:SOURce:VOLTage<i>

N (see [page 1708](#))

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 909
 - [":POWer:SIGNals:CYCLes:HARMonics"](#) on page 910
 - [":POWer:SIGNals:CYCLes:QUALity"](#) on page 911
 - [":POWer:SIGNals:DURation:EFFiciency"](#) on page 912
 - [":POWer:SIGNals:DURation:MODulation"](#) on page 913
 - [":POWer:SIGNals:DURation:ONOff:OFF"](#) on page 914
 - [":POWer:SIGNals:DURation:ONOff:ON"](#) on page 915
 - [":POWer:SIGNals:DURation:RIPple"](#) on page 916
 - [":POWer:SIGNals:DURation:TRANsient"](#) on page 917
 - [":POWer:SIGNals:IEXpected"](#) on page 918
 - [":POWer:SIGNals:OVERshoot"](#) on page 919
 - [":POWer:SIGNals:VMAXimum:INRush"](#) on page 920
 - [":POWer:SIGNals:VMAXimum:ONOff:OFF"](#) on page 921
 - [":POWer:SIGNals:VMAXimum:ONOff:ON"](#) on page 922
 - [":POWer:SIGNals:VSTeady:ONOff:OFF"](#) on page 923
 - [":POWer:SIGNals:VSTeady:ONOff:ON"](#) on page 924
 - [":POWer:SIGNals:VSTeady:TRANsient"](#) on page 925
 - [":POWer:SIGNals:SOURce:CURREnt<i>"](#) on page 926

:POWer:SLEW:APPLy

N (see [page 1708](#))

Command Syntax :POWer:SLEW:APPLy

The :POWer:SLEW:APPLy command applies the slew rate analysis.

See Also • [":POWer:SLEW:SOURce"](#) on page 929

:POWer:SLEW:SOURce

N (see [page 1708](#))

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 928

:POWer:SWITCh:APPLy

N (see [page 1708](#))

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 931
 - [":POWer:SWITCh:IREFERENCE"](#) on page 932
 - [":POWer:SWITCh:RDS"](#) on page 933
 - [":POWer:SWITCh:VCE"](#) on page 934
 - [":POWer:SWITCh:VREFERENCE"](#) on page 935
 - [":MEASure:ELOSs"](#) on page 779
 - [":MEASure:PLOSs"](#) on page 786

:POWer:SWITCh:CONDUction

N (see [page 1708](#))

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 930
 - [":POWer:SWITCh:IREFERENCE"](#) on page 932
 - [":POWer:SWITCh:RDS"](#) on page 933
 - [":POWer:SWITCh:VCE"](#) on page 934
 - [":POWer:SWITCh:VREFERENCE"](#) on page 935

:POWer:SWITCh:IREFERENCE

N (see [page 1708](#))

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 930
 - [":POWer:SWITCh:CONDUction"](#) on page 931
 - [":POWer:SWITCh:RDS"](#) on page 933
 - [":POWer:SWITCh:VCE"](#) on page 934
 - [":POWer:SWITCh:VREFERENCE"](#) on page 935

:POWer:SWITCh:RDS

N (see [page 1708](#))

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 930
 - [":POWer:SWITCh:CONDUction"](#) on page 931
 - [":POWer:SWITCh:IREFERENCE"](#) on page 932
 - [":POWer:SWITCh:VCE"](#) on page 934
 - [":POWer:SWITCh:VREFERENCE"](#) on page 935

:POWer:SWITCh:VCE

N (see [page 1708](#))

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 930
 - [":POWer:SWITCh:CONDUction"](#) on page 931
 - [":POWer:SWITCh:IREFERENCE"](#) on page 932
 - [":POWer:SWITCh:RDS"](#) on page 933
 - [":POWer:SWITCh:VREFERENCE"](#) on page 935

:POWer:SWITCh:VREference

N (see [page 1708](#))

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 930
 - [":POWer:SWITCh:CONDUCTION"](#) on page 931
 - [":POWer:SWITCh:IREference"](#) on page 932
 - [":POWer:SWITCh:RDS"](#) on page 933
 - [":POWer:SWITCh:VCE"](#) on page 934

:POWer:TRANsient:APPLy

N (see [page 1708](#))

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 937
 - [":POWer:TRANsient:IInitial"](#) on page 938
 - [":POWer:TRANsient:INew"](#) on page 939
 - [":POWer:TRANsient:NEXT"](#) on page 940
 - [":MEASure:TRESponse"](#) on page 791

:POWer:TRANsient:EXIT

N (see [page 1708](#))

Command Syntax :POWer:TRANsient:EXIT

The :POWer:TRANsient:EXIT command exits (stops) the transient analysis.

This command is equivalent to selecting **Exit** in the user interface during the analysis.

- See Also**
- [":POWer:TRANsient:APPLy"](#) on page 936
 - [":POWer:TRANsient:INItial"](#) on page 938
 - [":POWer:TRANsient:INEW"](#) on page 939
 - [":POWer:TRANsient:NEXT"](#) on page 940

:POWer:TRANsient:IINitial

N (see [page 1708](#))

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 925
 - [":POWer:TRANsient:APPLy"](#) on page 936
 - [":POWer:TRANsient:EXIT"](#) on page 937
 - [":POWer:TRANsient:INew"](#) on page 939
 - [":POWer:TRANsient:NEXT"](#) on page 940

:POWer:TRANsient:INEW

N (see [page 1708](#))

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 936
 - [":POWer:TRANsient:EXIT"](#) on page 937
 - [":POWer:TRANsient:IInitial"](#) on page 938
 - [":POWer:TRANsient:NEXT"](#) on page 940

:POWer:TRANsient:NEXT

N (see [page 1708](#))

Command Syntax :POWer:TRANsient:NEXT

The :POWer:TRANsient:NEXT command goes to the next step of the transient analysis.

This command is equivalent to selecting **Next** in the user interface when prompted during the analysis.

- See Also
- [":POWer:TRANsient:APPLY"](#) on page 936
 - [":POWer:TRANsient:EXIT"](#) on page 937
 - [":POWer:TRANsient:IInitial"](#) on page 938
 - [":POWer:TRANsient:INEW"](#) on page 939

33 :RECall Commands

Recall previously saved oscilloscope setups, reference waveforms, and masks.

Table 115 :RECall Commands Summary

Command	Query	Options and Query Returns
:RECall:ARBitrary:[ST ARt] [{<internal_fname> <external_fname>}] [, <column>] [, <wavegen_id>] (see page 944)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected <column> ::= Column in CSV file to load. Column number starts from 1. <wavegen_id> ::= WGEN1
:RECall:DBC[:START] [<file_name>] [, <protocol_decode_bus>] (see page 945)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".dbc". <protocol_decode_bus> ::= {PBUS<n> SBUS<n>} <n> ::= 1 to (# of protocol decode bus) in NR1 format
:RECall:FIleName <base_name> (see page 946)	:RECall:FIleName? (see page 946)	<base_name> ::= quoted ASCII string

Table 115 :RECall Commands Summary (continued)

Command	Query	Options and Query Returns
:RECall:LDF[:START] [<file_name>] [, <protocol_decode_bus>] (see page 947)	n/a	<file_name> ::= quoted ASCII string If extension included in file name, it must be ".ldf". <protocol_decode_bus> ::= {PBUS<n> SBUS<n>} <n> ::= 1 to (# of protocol decode bus) in NR1 format
:RECall:MASK[:START] [<internal_fname> <external_fname>]] (see page 948)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:RECall:PWD <path_name> (see page 949)	:RECall:PWD? (see page 949)	<path_name> ::= quoted ASCII string
:RECall:SETup[:START] [<internal_fname> <external_fname>]] (see page 950)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:RECall:WMEemory<r>[:START] [<file_name> <data>] (see page 951)	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". <data> ::= binary block data in IEEE 488.2 # format

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"
```

Recalling Files From a USB Storage Device

When :RECall commands have a "quoted ASCII string" <file_name> parameter, you can recall files from a connected USB storage device. For example:

```
' To recall a setup file from a connected USB storage device:  
myScope.WriteString ":RECall:SETup:START ""/usb/my_setup_file.scp"""
```

:RECall:ARbitrary[:START]

N (see [page 1708](#))

Command Syntax :RECall:ARbitrary[:START]
 [{<internal_fname> | <external_fname>}] [, <column>] [, <wavegen_id>]

 <internal_fname> ::= quoted ASCII file name string beginning
 with "/User Files/"

 <external_fname> ::= quoted ASCII file name string beginning
 with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending
 on the number of USB storage devices connected

 <column> ::= Column in CSV file to load. Column number starts from 1.
 <wavegen_id> ::= WGEN1 - specifies which wavegen

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 be 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 942
 - [":RECall:FILENAME"](#) on page 946
 - [":RECall:PWD"](#) on page 949
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:RECall:DBC[:START]

N (see [page 1708](#))

Command Syntax :RECall:DBC[:START] [<file_name>] [, <protocol_decode_bus>]

<file_name> ::= quoted ASCII string

<protocol_decode_bus> ::= {PBUS<n> | SBUS<n>}

<n> ::= 1 to (# of protocol decode 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 <protocol_decode_bus> parameter specifies which protocol decode waveform the CAN symbolic data will be loaded for.

- See Also**
- ["Introduction to :RECall Commands"](#) on page 942
 - [":RECall:FILENAME"](#) on page 946
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:MESSage"](#) on page 1025
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:SIGNal"](#) on page 1026
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:VALue"](#) on page 1027
 - [":SEARCH:SERial:CAN:MODE"](#) on page 1211
 - [":SEARCH:SERial:CAN:SYMBolic:MESSage"](#) on page 1217
 - [":SEARCH:SERial:CAN:SYMBolic:SIGNal"](#) on page 1218
 - [":SEARCH:SERial:CAN:SYMBolic:VALue"](#) on page 1219
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:RECall:FIlename

N (see [page 1708](#))

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 942
 - [":RECall:SETup\[:START\]"](#) on page 950
 - [":SAVE:FIlename"](#) on page 958

:RECall:LDF[:START]

N (see [page 1708](#))

Command Syntax :RECall:LDF[:START] [<file_name>] [, <protocol_decode_bus>]

<file_name> ::= quoted ASCII string

<protocol_decode_bus> ::= {PBUS<n> | SBUS<n>}

<n> ::= 1 to (# of protocol decode 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 <protocol_decode_bus> parameter specifies which protocol decode waveform the LIN symbolic data will be loaded for.

- See Also**
- ["Introduction to :RECall Commands"](#) on page 942
 - [":RECall:FILENAME"](#) on page 946
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:TRIGger:SYMBOLic:FRAME"](#) on page 1056
 - [":SBUS<n>:LIN:TRIGger:SYMBOLic:SIGNAL"](#) on page 1057
 - [":SBUS<n>:LIN:TRIGger:SYMBOLic:VALue"](#) on page 1058
 - [":SEARCH:SERial:LIN:MODE"](#) on page 1229
 - [":SEARCH:SERial:LIN:SYMBOLic:FRAME"](#) on page 1233
 - [":SEARCH:SERial:LIN:SYMBOLic:SIGNAL"](#) on page 1234
 - [":SEARCH:SERial:LIN:SYMBOLic:VALue"](#) on page 1235
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:RECall:MASK[:START]

N (see [page 1708](#))

Command Syntax :RECall:MASK[:START] [{<internal_fname> | <external_fname>}]

<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected

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".

If no filename parameter is included, the name specified by the :RECall:FILENAME command and the location specified by the :RECall:PWD command are used.

NOTE

When a USB storage device is connected, the :RECall:PWD location becomes "/usb/". When all USB storage devices are disconnected, the :RECall:PWD location becomes "/User Files/".

- See Also**
- ["Introduction to :RECall Commands"](#) on page 942
 - [":RECall:FILENAME"](#) on page 946
 - [":RECall:PWD"](#) on page 949
 - [":SAVE:MASK\[:START\]"](#) on page 963
 - [":MTEST:DATA"](#) on page 809

:RECall:PWD

N (see [page 1708](#))

Command Syntax :RECall:PWD <path_name>

<path_name> ::= quoted ASCII string

The :RECall:PWD command sets the present working directory for recall operations.

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 942
 - [":SAVE:PWD"](#) on page 966

:RECall:SETup[:START]

N (see [page 1708](#))

Command Syntax :RECall:SETup[:START] [{<internal_fname> | <external_fname>}]

<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected

The :RECall:SETup[:START] command recalls an oscilloscope setup from a file.

NOTE

If a file extension is provided as part of a specified <file_name>, it must be ".scp".

If no filename parameter is included, the name specified by the :RECall:FIleName command and the location specified by the :RECall:PWD command are used.

NOTE

When a USB storage device is connected, the :RECall:PWD location becomes "/usb/". When all USB storage devices are disconnected, the :RECall:PWD location becomes "/User Files/".

- See Also**
- ["Introduction to :RECall Commands"](#) on page 942
 - [":RECall:FIleName"](#) on page 946
 - [":RECall:PWD"](#) on page 949
 - [":SAVE:SETup\[:START\]"](#) on page 974
 - ["*RCL \(Recall\)"](#) on page 246

:RECall:WMEemory<r>[:START]

N (see [page 1708](#))

Command Syntax :RECall:WMEemory<r>[:START] [<file_name> | <data>]
 <r> ::= 1 to (# ref waveforms) in NR1 format
 <file_name> ::= quoted ASCII string
 <data> ::= binary block data in IEEE 488.2 # format

The :RECall:WMEemory<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".

The <data> option lets you recall a reference waveform from a local file on the controller PC (instead of from a USB storage device connected to the oscilloscope). In this case, your remote program reads data from the local ".h5" format reference waveform file in the same way that setup files are restored to the oscilloscope (see [":SYSTEM:SETup"](#) on page 1457).

- See Also**
- ["Introduction to :RECall Commands"](#) on page 942
 - [":RECall:FILENAME"](#) on page 946
 - [":SAVE:WMEemory\[:START\]"](#) on page 981
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

34 :SAVE Commands

Save oscilloscope setups, screen images, and data. See ["Introduction to :SAVE Commands"](#) on page 956.

Table 116 :SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:ARBitrary:[STARt] [{<internal_fname> <external_fname>}] [, <wavegen_id>] (see page 957)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected <wavegen_id> ::= WGEN1
:SAVE:FIleName <base_name> (see page 958)	:SAVE:FIleName? (see page 958)	<base_name> ::= quoted ASCII string
:SAVE:IMAGe[:STARt] [<file_name>] (see page 959)	n/a	<file_name> ::= quoted ASCII string
:SAVE:IMAGe:FACTors { {0 OFF} {1 ON} } (see page 960)	:SAVE:IMAGe:FACTors? (see page 960)	{0 1}
:SAVE:IMAGe:FORMat <format> (see page 961)	:SAVE:IMAGe:FORMat? (see page 961)	<format> ::= { {BMP BMP24bit} PNG NONE }
:SAVE:LISter[:STARt] [<file_name>] (see page 962)	n/a	<file_name> ::= quoted ASCII string

Table 116 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:MASK[:START] [{<internal_fname> <external_fname>}] (see page 963)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:SAVE:MULTi[:START] [<file_name>] (see page 964)	n/a	<file_name> ::= quoted ASCII string
:SAVE:POWer:[START] [<file_spec>] (see page 965)	n/a	<file_name> ::= quoted ASCII string
:SAVE:PWD <path_name> (see page 966)	:SAVE:PWD? (see page 966)	<path_name> ::= quoted ASCII string
:SAVE:RESults:[START] [<file_spec>] (see page 967)	n/a	<file_name> ::= quoted ASCII string
:SAVE:RESults:FORMat: CURSor {{0 OFF} {1 ON}} (see page 968)	:SAVE:RESults:FORMat: CURSor? (see page 968)	{0 1}
:SAVE:RESults:FORMat: HISTogram {{0 OFF} {1 ON}} (see page 969)	:SAVE:RESults:FORMat: HISTogram? (see page 969)	{0 1}
:SAVE:RESults:FORMat: MASK {{0 OFF} {1 ON}} (see page 970)	:SAVE:RESults:FORMat: MASK? (see page 970)	{0 1}
:SAVE:RESults:FORMat: MEASurement {{0 OFF} {1 ON}} (see page 971)	:SAVE:RESults:FORMat: MEASurement? (see page 971)	{0 1}
:SAVE:RESults:FORMat: SEARch {{0 OFF} {1 ON}} (see page 972)	:SAVE:RESults:FORMat: SEARch? (see page 972)	{0 1}

Table 116 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:RESults:FORMat:SEGmented {{0 OFF} {1 ON}} (see page 973)	:SAVE:RESults:FORMat:SEGmented? (see page 973)	{0 1}
:SAVE:SETup[:START] [{<internal_fname> <external_fname>}] (see page 974)	n/a	<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/" <external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected
:SAVE:WAVEform[:START] [<file_name>] (see page 975)	n/a	<file_name> ::= quoted ASCII string
:SAVE:WAVEform:FORMat <format> (see page 976)	:SAVE:WAVEform:FORMat ? (see page 976)	<format> ::= {ASCIixy CSV BINary NONE}
:SAVE:WAVEform:LENGth <length> (see page 977)	:SAVE:WAVEform:LENGth ? (see page 977)	<length> ::= 100 to max. length; an integer in NR1 format
:SAVE:WAVEform:LENGth:MAX {{0 OFF} {1 ON}} (see page 978)	:SAVE:WAVEform:LENGth:MAX? (see page 978)	{0 1}
:SAVE:WAVEform:SEGMen ted <option> (see page 979)	:SAVE:WAVEform:SEGMen ted? (see page 979)	<option> ::= {ALL CURRent}

Table 116 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
:SAVE:WMEemory:SOURce <source> (see page 980)	:SAVE:WMEemory:SOURce? (see page 980)	<source> ::= {CHANnel<n> FUNCTION<m> MATH<m> WMEemory<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:WMEemory[:START] [<file_name>] (see page 981)	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 "scope_0";:SAVE:IMAG:FACT 0;FORM NONE;  
:SAVE:PWD "/User Files/setups/";:SAVE:WAV:FORM NONE;LENG +2000;  
SEGM CURR
```

Saving Files to a USB Storage Device

When :SAVE commands have a "quoted ASCII string" <file_name> parameter, you can save files to a connected USB storage device. For example:

```
' To save a setup file to a connected USB storage device:  
myScope.WriteString ":SAVE:SETup:START ""/usb/my_setup_file.scp"""
```


:SAVE:ARbitrary[:START]

N (see [page 1708](#))

Command Syntax :SAVE:ARbitrary[:START]
 [{<internal_fname> | <external_fname>}] [, <wavegen_id>]

 <internal_fname> ::= quoted ASCII file name string beginning
 with "/User Files/"

 <external_fname> ::= quoted ASCII file name string beginning
 with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending
 on the number of USB storage devices connected

 <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".

If no filename parameter is included, the name specified by the :SAVE:FILENAME command and the location specified by the :SAVE:PWD command are used.

NOTE

When a USB storage device is connected, the :SAVE:PWD location becomes "/usb/". When all USB storage devices are disconnected, the :SAVE:PWD location becomes "/User Files/".

- See Also**
- ["Introduction to :SAVE Commands"](#) on page 956
 - [":SAVE:FILENAME"](#) on page 958
 - [":SAVE:PWD"](#) on page 966
 - [":RECall:ARbitrary\[:START\]"](#) on page 944
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:SAVE:FILENAME

N (see [page 1708](#))

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 956
 - [":SAVE:IMAGe\[:START\]"](#) on page 959
 - [":SAVE:SETup\[:START\]"](#) on page 974
 - [":SAVE:WAVEform\[:START\]"](#) on page 975
 - [":SAVE:PWD"](#) on page 966
 - [":RECall:FILENAME"](#) on page 946

:SAVE:IMAGe[:START]

N (see [page 1708](#))

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 956
 - [":SAVE:IMAGe:FACTors"](#) on page 960
 - [":SAVE:IMAGe:FORMat"](#) on page 961
 - [":SAVE:FILEname"](#) on page 958

:SAVE:IMAGe:FACTors

N (see [page 1708](#))

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 956
 - [":SAVE:IMAGe\[:START\]"](#) on page 959
 - [":SAVE:IMAGe:FORMat"](#) on page 961

:SAVE:IMAGe:FORMat

N (see [page 1708](#))

Command Syntax :SAVE:IMAGe:FORMat <format>

<format> ::= {{BMP | BMP24bit} | 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 | PNG | NONE}

When NONE is returned, it indicates that a non-image save format is currently selected.

- See Also**
- ["Introduction to :SAVE Commands"](#) on page 956
 - [":SAVE:IMAGe\[:START\]"](#) on page 959
 - [":SAVE:IMAGe:FACTors"](#) on page 960
 - [":SAVE:WAVEform:FORMat"](#) on page 976

:SAVE:LISTer[:START]

N (see [page 1708](#))

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 956
 - [":SAVE:FILENAME"](#) on page 958
 - [Chapter 25](#), ":LISTer Commands," starting on page 613

:SAVE:MASK[:START]

N (see [page 1708](#))

Command Syntax :SAVE:MASK[:START] [{<internal_fname> | <external_fname>}]

<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected

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".

If no filename parameter is included, the name specified by the :SAVE:FILENAME command and the location specified by the :SAVE:PWD command are used.

NOTE

When a USB storage device is connected, the :SAVE:PWD location becomes "/usb/". When all USB storage devices are disconnected, the :SAVE:PWD location becomes "/User Files/".

- See Also**
- ["Introduction to :SAVE Commands"](#) on page 956
 - [":SAVE:FILENAME"](#) on page 958
 - [":SAVE:PWD"](#) on page 966
 - [":RECALL:MASK\[:START\]"](#) on page 948
 - [":MTEST:DATA"](#) on page 809

:SAVE:MULTi[:START]

N (see [page 1708](#))

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 Infiniium 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 956
 - [":SAVE:FILENAME"](#) on page 958
 - [":SAVE:PWD"](#) on page 966

:SAVE:POWer[:START]

N (see [page 1708](#))

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 956
 - [":SAVE:FILENAME"](#) on page 958
 - [Chapter 32](#), ":POWer Commands," starting on page 837

:SAVE:PWD

N (see [page 1708](#))

Command Syntax :SAVE:PWD <path_name>

<path_name> ::= quoted ASCII string

The :SAVE:PWD command sets the present working directory for save operations.

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 956
 - [":SAVE:FILENAME"](#) on page 958
 - [":RECall:PWD"](#) on page 949

:SAVE:RESults[:START]

N (see [page 1708](#))

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 968
 - [":SAVE:RESults:FORMat:MASK"](#) on page 970
 - [":SAVE:RESults:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESults:FORMat:SEARCh"](#) on page 972
 - [":SAVE:RESults:FORMat:SEGmented"](#) on page 973

:SAVE:RESuLts:FORMat:CURSor

N (see [page 1708](#))

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 967
 - [":SAVE:RESuLts:FORMat:MASK"](#) on page 970
 - [":SAVE:RESuLts:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESuLts:FORMat:SEARCh"](#) on page 972
 - [":SAVE:RESuLts:FORMat:SEGMENTed"](#) on page 973

:SAVE:RESuLts:FORMat:HISTogram

N (see [page 1708](#))

Command Syntax :SAVE:RESuLts:FORMat:HISTogram {{0 | OFF} | {1 | ON}}

The :SAVE:RESuLts:FORMat:HISTogram command specifies whether histogram 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:HISTogram?

The :SAVE:RESuLts:FORMat:HISTogram? query returns whether histogram statistics will be included when analysis results are saved.

Return Format <off_on><NL>
{0 | 1}

- See Also**
- [":SAVE:RESuLts:\[START\]"](#) on page 967
 - [":SAVE:RESuLts:FORMat:CURSor"](#) on page 968
 - [":SAVE:RESuLts:FORMat:MASK"](#) on page 970
 - [":SAVE:RESuLts:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESuLts:FORMat:SEARch"](#) on page 972
 - [":SAVE:RESuLts:FORMat:SEGmented"](#) on page 973

:SAVE:RESuLts:FORMat:MASK

N (see [page 1708](#))

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 967
 - [":SAVE:RESuLts:FORMat:CURSor"](#) on page 968
 - [":SAVE:RESuLts:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESuLts:FORMat:SEARch"](#) on page 972
 - [":SAVE:RESuLts:FORMat:SEGMENTed"](#) on page 973

:SAVE:RESuLts:FORMat:MEASurement

N (see [page 1708](#))

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 967
 - [":SAVE:RESuLts:FORMat:CURSor"](#) on page 968
 - [":SAVE:RESuLts:FORMat:MASK"](#) on page 970
 - [":SAVE:RESuLts:FORMat:SEARCh"](#) on page 972
 - [":SAVE:RESuLts:FORMat:SEGMented"](#) on page 973

:SAVE:RESuLts:FORMat:SEARCh

N (see [page 1708](#))

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 967
 - [":SAVE:RESuLts:FORMat:CURSOr"](#) on page 968
 - [":SAVE:RESuLts:FORMat:MASK"](#) on page 970
 - [":SAVE:RESuLts:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESuLts:FORMat:SEGMented"](#) on page 973

:SAVE:RESuLts:FORMat:SEGmented

N (see [page 1708](#))

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 967
 - [":SAVE:RESuLts:FORMat:CURSor"](#) on page 968
 - [":SAVE:RESuLts:FORMat:MASK"](#) on page 970
 - [":SAVE:RESuLts:FORMat:MEASurement"](#) on page 971
 - [":SAVE:RESuLts:FORMat:SEARch"](#) on page 972

:SAVE:SETup[:START]

N (see [page 1708](#))

Command Syntax :SAVE:SETup[:START] [{<internal_fname> | <external_fname>}]

<internal_fname> ::= quoted ASCII file name string beginning with "/User Files/"

<external_fname> ::= quoted ASCII file name string beginning with "/USB1/" (or "/usb/"), "/USB2/", or "/USB3/" depending on the number of USB storage devices connected

The :SAVE:SETup[:START] command saves the oscilloscope setup to a file.

NOTE

If a file extension is provided as part of a specified file name, it must be ".scp".

If no filename parameter is included, the name specified by the :SAVE:FILENAME command and the location specified by the :SAVE:PWD command are used.

NOTE

When a USB storage device is connected, the :SAVE:PWD location becomes "/usb/". When all USB storage devices are disconnected, the :SAVE:PWD location becomes "/User Files/".

- See Also**
- ["Introduction to :SAVE Commands"](#) on page 956
 - [":SAVE:FILENAME"](#) on page 958
 - [":SAVE:PWD"](#) on page 966
 - [":RECall:SETup\[:START\]"](#) on page 950
 - ["*SAV \(Save\)"](#) on page 250

:SAVE:WAVEform[:START]

N (see [page 1708](#))

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 956
 - [":SAVE:WAVEform:FORMat"](#) on page 976
 - [":SAVE:WAVEform:LENGth"](#) on page 977
 - [":SAVE:FILEname"](#) on page 958
 - [":RECall:SETup\[:START\]"](#) on page 950

:SAVE:WAVEform:FORMat

N (see [page 1708](#))

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 a non-waveform save format is currently selected.

- See Also**
- ["Introduction to :SAVE Commands"](#) on page 956
 - [":SAVE:WAVEform\[:START\]"](#) on page 975
 - [":SAVE:WAVEform:LENGth"](#) on page 977
 - [":SAVE:IMAGe:FORMat"](#) on page 961

:SAVE:WAVEform:LENGth

N (see [page 1708](#))

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 956
 - [":SAVE:WAVEform:LENGth:MAX"](#) on page 978
 - [":SAVE:WAVEform\[:START\]"](#) on page 975
 - [":WAVEform:POINts"](#) on page 1566
 - [":SAVE:WAVEform:FORMat"](#) on page 976

:SAVE:WAVEform:LENGth:MAX

N (see [page 1708](#))

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 956
 - [":SAVE:WAVEform\[:START\]"](#) on page 975
 - [":SAVE:WAVEform:LENGth"](#) on page 977

:SAVE:WAVEform:SEGMented

N (see [page 1708](#))

Command Syntax :SAVE:WAVEform:SEGMented <option>

<option> ::= {ALL | CURRent}

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.
- CURRent – only the currently selected segment is saved.

NOTE

This command gives a -221, "Settings conflict" error if a waveform save format has not been selected with the :SAVE:WAVEform:FORMat command.

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 956
 - [":SAVE:WAVEform\[:START\]"](#) on page 975
 - [":SAVE:WAVEform:FORMat"](#) on page 976
 - [":SAVE:WAVEform:LENGth"](#) on page 977

:SAVE:WMEemory:SOURce

N (see [page 1708](#))

Command Syntax :SAVE:WMEemory:SOURce <source>

<source> ::= {CHANnel<n> | FUNCTION<m> | MATH<m> | WMEemory<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:WMEemory: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:WMEemory:SOURce?

The :SAVE:WMEemory: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 956
 - [":SAVE:WMEemory\[:START\]"](#) on page 981
 - [":RECall:WMEemory<r>\[:START\]"](#) on page 951

:SAVE:WMEemory[:START]

N (see [page 1708](#))

Command Syntax :SAVE:WMEemory[:START] [<file_name>]

<file_name> ::= quoted ASCII string

The :SAVE:WMEemory[: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 956
 - [":SAVE:WMEemory:SOURce"](#) on page 980
 - [":RECall:WMEemory<r>\[:START\]"](#) on page 951

35 :SBUS<n> Commands

Control the modes and parameters for each serial bus decode/trigger type. See:

- **"Introduction to :SBUS<n> Commands"** on page 983
- **"General :SBUS<n> Commands"** on page 985
- **":SBUS<n>:A429 Commands"** on page 988
- **":SBUS<n>:CAN Commands"** on page 1004
- **":SBUS<n>:IIC Commands"** on page 1031
- **":SBUS<n>:LIN Commands"** on page 1041
- **":SBUS<n>:M1553 Commands"** on page 1059
- **":SBUS<n>:SENT Commands"** on page 1066
- **":SBUS<n>:SPI Commands"** on page 1102
- **":SBUS<n>:UART Commands"** on page 1119
- **":SBUS<n>:USB Commands"** on page 1136
- **":SBUS<n>:USBPd Commands"** on page 1161

Introduction to :SBUS<n> Commands

The :SBUS subsystem commands control the serial decode bus viewing, mode, and other options.

NOTE

"PBUS" (protocol decode bus) is now an alias for "SBUS" (serial bus) because terminology has changed in the HD3-Series oscilloscope. The HD3-Series oscilloscope has a **[Protocol Decode]** front panel key and refers to the feature as "Protocol Decode/Triggering". Previous InfiniiVision X-Series oscilloscopes had a **[Serial]** front panel key and refer to the feature as "Serial Decode/Triggering".

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 1494).

- **ARINC 429 triggering** — lets you trigger on ARINC 429 serial data.

- **CAN (Controller Area Network) triggering** – will trigger on CAN 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.
- **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.
- **MIL-STD-1553 triggering** – lets you trigger on MIL-STD-1553 serial data.
- **SENT triggering** – lets you trigger on SENT serial 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** – lets you trigger on RS-232 serial data.
- **USB 2.0 triggering** – lets you trigger on USB 2.0 serial data.
- **USB PD triggering** – lets you trigger on USB PD serial data.

NOTE

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 117 General :SBUS<n> Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:DISPlay { {0 OFF} {1 ON} } (see page 986)	:SBUS<n>:DISPlay? (see page 986)	{0 1}
:SBUS<n>:MODE <mode> (see page 987)	:SBUS<n>:MODE? (see page 987)	<mode> ::= {A429 CAN IIC LIN M1553 SENT SPI UART USB USBPd}

:SBUS<n>:DISPlay

N (see [page 1708](#))

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 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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
• [":CHANnel<n>:DISPlay"](#) on page 351
• [":DIGital<d>:DISPlay"](#) on page 400
• [":POD<n>:DISPlay"](#) on page 831
• [":VIEW"](#) on page 281
• [":BLANK"](#) on page 270
• [":STATus?"](#) on page 278

:SBUS<n>:MODE

N (see [page 1708](#))

Command Syntax :SBUS<n>:MODE <mode>

<mode> ::= {A429 | CAN | IIC | LIN | M1553 | SENT | SPI | UART | USB
| USBPd}

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 | CAN | IIC | LIN | M1553 | SENT | SPI | UART | USB
| USBP | NONE}

Errors • ["-241, Hardware missing"](#) on page 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983

- [":SBUS<n>:A429 Commands"](#) on page 988
- [":SBUS<n>:CAN Commands"](#) on page 1004
- [":SBUS<n>:IIC Commands"](#) on page 1031
- [":SBUS<n>:LIN Commands"](#) on page 1041
- [":SBUS<n>:M1553 Commands"](#) on page 1059
- [":SBUS<n>:SENT Commands"](#) on page 1066
- [":SBUS<n>:SPI Commands"](#) on page 1102
- [":SBUS<n>:UART Commands"](#) on page 1119
- [":SBUS<n>:USB Commands"](#) on page 1136
- [":SBUS<n>:USBPd Commands"](#) on page 1161

:SBUS<n>:A429 Commands

NOTE

These commands are valid when the Aerospace and Defense Serial Analysis license has been enabled.

NOTE

Aerospace protocol decodes (ARINC 429 or MIL-STD-155) and Automotive protocol decodes (CAN, LIN, or SENT) cannot be used at the same time. Protocol decode mode choices on Protocol2 are limited by the last Aerospace or Automotive protocol decode mode selected on Protocol1.

Table 118 :SBUS<n>:A429 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:A429:AUTOset up (see page 990)	n/a	n/a
:SBUS<n>:A429:BASE <base> (see page 991)	:SBUS<n>:A429:BASE? (see page 991)	<base> ::= {BINary HEX}
:SBUS<n>:A429:BAUDrat e <baudrate> (see page 992)	:SBUS<n>:A429:BAUDrat e? (see page 992)	<baudrate> ::= integer from 10000 to 1000000
:SBUS<n>:A429:FORMat <format> (see page 993)	:SBUS<n>:A429:FORMat? (see page 993)	<format> ::= {LDSDi LDSSm LDATa}
:SBUS<n>:A429:SIGNal <signal> (see page 994)	:SBUS<n>:A429:SIGNal? (see page 994)	<signal> ::= {A B DIFFerential}
:SBUS<n>:A429:SOURce <source> (see page 995)	:SBUS<n>:A429:SOURce? (see page 995)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:A429:SPEEd <speed> (see page 996)	:SBUS<n>:A429:SPEEd? (see page 996)	<speed> ::= {LOW HIGH USER}
:SBUS<n>:A429:TRIGger :LABel <value> (see page 997)	:SBUS<n>:A429:TRIGger :LABel? (see page 997)	<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 118 :SBUS<n>:A429 Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:A429:TRIGger :PATtern:DATA <string> (see page 998)	:SBUS<n>:A429:TRIGger :PATtern:DATA? (see page 998)	<string> ::= "nn...n" where n ::= {0 1 X}, length depends on FORMat
:SBUS<n>:A429:TRIGger :PATtern:SDI <string> (see page 999)	:SBUS<n>:A429:TRIGger :PATtern:SDI? (see page 999)	<string> ::= "nn" where n ::= {0 1 X}, length always 2 bits
:SBUS<n>:A429:TRIGger :PATtern:SSM <string> (see page 1000)	:SBUS<n>:A429:TRIGger :PATtern:SSM? (see page 1000)	<string> ::= "nn" where n ::= {0 1 X}, length always 2 bits
:SBUS<n>:A429:TRIGger :RANGe <min>,<max> (see page 1001)	:SBUS<n>:A429:TRIGger :RANGe? (see page 1001)	<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 1002)	:SBUS<n>:A429:TRIGger :TYPE? (see page 1002)	<condition> ::= {WSTArt WSTOp LABel LBITs PERRor WERRor GERRor WGERRors ALLerrors LRange ABITs AOBits AZBits}

:SBUS<n>:A429:AUTosetup

N (see [page 1708](#))

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 1663

- See Also**
- [":SBUS<n>:A429:BASE"](#) on page 991
 - [":SBUS<n>:A429:FORMat"](#) on page 993
 - [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 - ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:A429 Commands"](#) on page 988

:SBUS<n>:A429:BASE

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
 • [":SBUS<n>:MODE"](#) on page 987
 • [":SBUS<n>:A429:FORMat"](#) on page 993

:SBUS<n>:A429:BAUDrate

N (see [page 1708](#))

Command Syntax :SBUS<n>:A429:BAUDrate <baudrate>

<baudrate> ::= integer from 10000 to 1000000

When a user-defined baud rate is selected (with the ":SBUS<n>:A429:SPEEd USER" command), the :SBUS<n>:A429:BAUDrate command specifies the user-defined baud rate. The baud rate can be set in 100 b/s increments between 10000 and 100000 and in 1000 b/s increments between 100000 and 1000000.

Query Syntax :SBUS<n>:A429:BAUDrate?

The :SBUS<n>:A429:BAUDrate? query returns the user-defined baud rate setting.

Return Format <baudrate><NL>

See Also • [":SBUS<n>:A429:SPEEd"](#) on page 996

:SBUS<n>:A429:FORMat

N (see [page 1708](#))

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 1663

- See Also**
- ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:A429:TRIGger:PATtern:DATA"](#) on page 998
 - [":SBUS<n>:A429:TRIGger:PATtern:SDI"](#) on page 999
 - [":SBUS<n>:A429:TRIGger:PATtern:SSM"](#) on page 1000
 - [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 - [":SBUS<n>:A429:SIGNal"](#) on page 994
 - [":SBUS<n>:A429:SPEed"](#) on page 996
 - [":SBUS<n>:A429:BASE"](#) on page 991
 - [":SBUS<n>:A429:SOURce"](#) on page 995

:SBUS<n>:A429:SIGNal

N (see [page 1708](#))

Command Syntax :SBUS<n>:A429:SIGNal <signal>

<signal> ::= {A | B | DIFFerential}

The :SBUS<n>: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<n>: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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983

- [":SBUS<n>:MODE"](#) on page 987
- [":SBUS<n>:A429:FORMat"](#) on page 993
- [":SBUS<n>:A429:SPEed"](#) on page 996
- [":SBUS<n>:A429:SOURce"](#) on page 995

:SBUS<n>:A429:SOURce

N (see [page 1708](#))

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 threshold levels for the selected source.

Return Format <source><NL>

- See Also**
- [":TRIGger:LEVel:HIGH"](#) on page 1492
 - [":TRIGger:LEVel:LOW"](#) on page 1493
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 - [":SBUS<n>:A429:SIGNal"](#) on page 994
 - [":SBUS<n>:A429:SPEed"](#) on page 996
 - [":SBUS<n>:A429:FORMat"](#) on page 993
 - ["Introduction to :TRIGger Commands"](#) on page 1479

:SBUS<n>:A429:SPEEd

N (see [page 1708](#))

Command Syntax :SBUS<n>:A429:SPEEd <speed>

<speed> ::= {LOW | HIGH | USER}

The :SBUS<n>:A429:SPEEd command specifies the signal speed:

- LOW – 12.5 kb/s.
- HIGH – 100 kb/s.
- USER – lets you specify a user-defined baud rate using the :SBUS<n>:A429:BAUDRate command.

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 | USER}

Errors • ["-241, Hardware missing"](#) on page 1663

- See Also**
- [":SBUS<n>:A429:BAUDRate"](#) on page 992
 - ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:A429:SIGNal"](#) on page 994
 - [":SBUS<n>:A429:FORMat"](#) on page 993
 - [":SBUS<n>:A429:SOURce"](#) on page 995

:SBUS<n>:A429:TRIGger:LABel

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
• [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002

:SBUS<n>:A429:TRIGger:PATtern:DATA

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 • [":SBUS<n>:A429:TRIGger:PATtern:SDI"](#) on page 999
 • [":SBUS<n>:A429:TRIGger:PATtern:SSM"](#) on page 1000

:SBUS<n>:A429:TRIGger:PATtern:SDI

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:A429:FORMat"](#) on page 993
 • [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 • [":SBUS<n>:A429:TRIGger:PATtern:DATA"](#) on page 998
 • [":SBUS<n>:A429:TRIGger:PATtern:SSM"](#) on page 1000

:SBUS<n>:A429:TRIGger:PATtern:SSM

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:A429:FORMat"](#) on page 993
 • [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002
 • [":SBUS<n>:A429:TRIGger:PATtern:DATA"](#) on page 998
 • [":SBUS<n>:A429:TRIGger:PATtern:SDI"](#) on page 999

:SBUS<n>:A429:TRIGger:RANGe

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
• [":SBUS<n>:A429:TRIGger:TYPE"](#) on page 1002

:SBUS<n>:A429:TRIGger:TYPE

N (see [page 1708](#))

Command Syntax :SBUS<n>:A429:TRIGger:TYPE <condition>

```
<condition> ::= {WSTArt | WSTOp | LABel | LBITs | PERRor | WERRor
                | GERRor | WGERrors | ALLerrors | LRANge | ABITs
                | AOBits | AZBits}
```

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.
- WGERrors – 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.

Query Syntax :SBUS<n>:A429:TRIGger:TYPE?

The :SBUS<n>:A429:TRIGger:TYPE? query returns the current ARINC 429 trigger on condition.

Return Format <condition><NL>

```
<condition> ::= {WSTA | WSTO | LAB | LBIT | PERR | WERR | GERR | WGER
                | ALL | LRAN | ABIT | AOB | AZB}
```

Errors • ["-241, Hardware missing"](#) on page 1663

- See Also**
- ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:A429:TRIGger:LABel"](#) on page 997
 - [":SBUS<n>:A429:TRIGger:PATtern:DATA"](#) on page 998
 - [":SBUS<n>:A429:TRIGger:PATtern:SDI"](#) on page 999
 - [":SBUS<n>:A429:TRIGger:PATtern:SSM"](#) on page 1000
 - [":SBUS<n>:A429:TRIGger:RANge"](#) on page 1001

- **":SBUS<n>:A429:SOURce"** on page 995

:SBUS<n>:CAN Commands

NOTE

These commands are valid when the Automotive Serial Analysis license has been enabled.

NOTE

Aerospace protocol decodes (ARINC 429 or MIL-STD-155) and Automotive protocol decodes (CAN, LIN, or SENT) cannot be used at the same time. Protocol decode mode choices on Protocol2 are limited by the last Aerospace or Automotive protocol decode mode selected on Protocol1.

Table 119 :SBUS<n>:CAN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:CAN:DISPlay <type> (see page 1007)	:SBUS<n>:CAN:DISPlay? (see page 1007)	<type> ::= {HEXadecimal SYMBolic}
:SBUS<n>:CAN:FDSPoint <value> (see page 1008)	:SBUS<n>:CAN:FDSPoint ? (see page 1008)	<value> ::= even numbered percentages from 30 to 90 in NR3 format.
:SBUS<n>:CAN:SAMPlEpo int <percent> (see page 1009)	:SBUS<n>:CAN:SAMPlEpo int? (see page 1009)	<percent> ::= 30.0 to 90.0 in NR3 format
:SBUS<n>:CAN:SIGNAL:B AUDrate <baudrate> (see page 1010)	:SBUS<n>:CAN:SIGNAL:B AUDrate? (see page 1010)	<baudrate> ::= integer from 10000 to 4000000 in 100 b/s increments, or 5000000
:SBUS<n>:CAN:SIGNAL:D EFinition <value> (see page 1011)	:SBUS<n>:CAN:SIGNAL:D EFinition? (see page 1011)	<value> ::= {CANH CANL RX TX DIFFerential DIPL DIFH}
:SBUS<n>:CAN:SIGNAL:F DBaudrate <baudrate> (see page 1012)	:SBUS<n>:CAN:SIGNAL:F DBaudrate? (see page 1012)	<baudrate> ::= integer from 10000 to 10000000 in 100 b/s increments.
:SBUS<n>:CAN:SIGNAL:X LBaudrate <baudrate> (see page 1013)	:SBUS<n>:CAN:SIGNAL:X LBaudrate? (see page 1013)	<baudrate> ::= integer from 10000 to 20000000 in 100 b/s increments.
:SBUS<n>:CAN:SOURce <source> (see page 1014)	:SBUS<n>:CAN:SOURce? (see page 1014)	<source> ::= {CHANnel<n> DIGital<d> } <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 119 :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:TRIGger<condition> (see page 1015)	:SBUS<n>:CAN:TRIGger? (see page 1016)	<condition> ::= {SOF EOF IDData DATA IDRemote IDEither ERROR ACKerror FORMerror STUFFerror CRCerror SPECerror ALLerrors BRSBit CRCDbit EBActive EBPAssive OVERload MESSage MSIGnal}
:SBUS<n>:CAN:TRIGger:IDFilter {{0 OFF} {1 ON}} (see page 1018)	:SBUS<n>:CAN:TRIGger:IDFilter? (see page 1018)	{0 1}
:SBUS<n>:CAN:TRIGger:PATtern:DATA <string> (see page 1019)	:SBUS<n>:CAN:TRIGger:PATtern:DATA? (see page 1019)	<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 1020)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:DLC? (see page 1020)	<dlc> ::= integer between -1 (don't care) and 64, in NR1 format.
:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth<length> (see page 1021)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth? (see page 1021)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:CAN:TRIGger:PATtern:DATA:START<start> (see page 1022)	:SBUS<n>:CAN:TRIGger:PATtern:DATA:START? (see page 1022)	<start> ::= integer between 0 and 63, in NR1 format.
:SBUS<n>:CAN:TRIGger:PATtern:ID <string> (see page 1023)	:SBUS<n>:CAN:TRIGger:PATtern:ID? (see page 1023)	<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 1024)	:SBUS<n>:CAN:TRIGger:PATtern:ID:MODE? (see page 1024)	<value> ::= {STANdard EXTENDED}
:SBUS<n>:CAN:TRIGger:SYMBolic:MESSage<name> (see page 1025)	:SBUS<n>:CAN:TRIGger:SYMBolic:MESSage? (see page 1025)	<name> ::= quoted ASCII string
:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAL<name> (see page 1026)	:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNAL? (see page 1026)	<name> ::= quoted ASCII string

Table 119 :SBUS<n>:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue <data> (see page 1027)	:SBUS<n>:CAN:TRIGger:SYMBOLic:VALue? (see page 1027)	<data> ::= value in NR3 format
:SBUS<n>:CAN:TRIGger:TYPE <type> (see page 1028)	:SBUS<n>:CAN:TRIGger:TYPE? (see page 1028)	<type> ::= {STANDARD FD XL}
:SBUS<n>:CAN:TYPE <type> (see page 1029)	:SBUS<n>:CAN:TYPE? (see page 1029)	<type> ::= {STANDARD FD XFAST XSIC}
:SBUS<n>:CAN:XLSPoint <value> (see page 1030)	:SBUS<n>:CAN:XLSPoint? (see page 1030)	<value> ::= even numbered percentages from 30 to 90 in NR3 format.

:SBUS<n>:CAN:DISPlay

N (see [page 1708](#))

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 945

:SBUS<n>:CAN:FDSPoint

N (see [page 1708](#))

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 1012
 - [":SBUS<n>:CAN:TYPE"](#) on page 1029

:SBUS<n>:CAN:SAMPlEpoint

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015

:SBUS<n>:CAN:SIGNal:BAUDRate

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:SIGNal:DEFinition"](#) on page 1011
 - [":SBUS<n>:CAN:SOURce"](#) on page 1014

:SBUS<n>:CAN:SIGNal:DEFinition

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:CAN:SIGNal:BAUDrate"](#) on page 1010
 - [":SBUS<n>:CAN:SOURce"](#) on page 1014
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015

:SBUS<n>:CAN:SIGNal:FDBaudrate

N (see [page 1708](#))

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 1008
 - [":SBUS<n>:CAN:SIGNal:BAUDrate"](#) on page 1010
 - [":SBUS<n>:CAN:TYPE"](#) on page 1029

:SBUS<n>:CAN:SIGNal:XLBaudrate

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN:SIGNal:XLBaudrate <baudrate>

<baudrate> ::= integer from 10000 to 20000000 in 100 b/s increments.

The :SBUS<n>:CAN:SIGNal:XLBaudrate command sets the CAN XL baud rate from 10 kb/s to 20 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 XL, both the standard rate settings (see :SBUS<n>:CAN:SIGNal:BAUDrate), the FD rate settings, and the XL rate settings must be set correctly; otherwise, false triggers may occur.

Query Syntax :SBUS<n>:CAN:SIGNal:XLBaudrate?

The :SBUS<n>:CAN:SIGNal:XLBaudrate? query returns the current CAN XL baud rate setting.

Return Format <baudrate><NL>

<baudrate> ::= integer from 10000 to 20000000 in 100 b/s increments.

- See Also**
- [":SBUS<n>:CAN:XLSPoint"](#) on page 1030
 - [":SBUS<n>:CAN:SIGNal:BAUDrate"](#) on page 1010
 - [":SBUS<n>:CAN:SIGNal:FDbaudrate"](#) on page 1012
 - [":SBUS<n>:CAN:TYPE"](#) on page 1029

:SBUS<n>:CAN:SOURce

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN: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>: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 1479
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:SIGNAL:DEFinition"](#) on page 1011

:SBUS<n>:CAN:TRIGger

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN:TRIGger <condition>

```
<condition> ::= {SOF | EOF | IDData | DATA | IDRemote
| IDEither | ERRor | ACKerror | FORMerror | STUFferror | CRCerror
| SPECerror | ALLerrors | BRSBit | CRCDbit | EBActive | EBPAssive
| OVERload | MESSage | MSIGnal}
```

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.	
DATA	Data Frame ID and Data (non-FD)	With the STANdard trigger type (see :SBUS<n>:CAN:TRIGger:TYPE), 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. With the FD or XL trigger types (see :SBUS<n>:CAN:TRIGger:TYPE), triggers on CAN FD frames at the end of the last data byte defined in the trigger. You can trigger on up to 8 bytes of data anywhere within the CAN FD data, which can be up to 64 bytes long.	
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
FORMerror	Form Error	Triggers on reserved bit errors.	X

Condition	Front-panel name	Description	Filter by ID*
STUFFerror	Stuff Error	Triggers on 6 consecutive 1s or 6 consecutive 0s, while in a non-error or non overload frame.	X
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
EBActive	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)	With the STANDARD trigger type (see :SBUS<n>:CAN:TRIGGER:TYPE), triggers on a symbolic message and a signal value. With the FD or XL trigger types (see :SBUS<n>:CAN:TRIGGER:TYPE), 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 | IDR | IDE | ERR | ACK
                | FORM | STUF | CRC | SPEC | ALL | BRSB | CRCD | EBA | EBP | OVER
                | MESS | MSIG}

```

- Errors
- ["-241, Hardware missing"](#) on page 1663
- See Also
- [":SBUS<n>:CAN:TRIGger:TYPE"](#) on page 1028
 - ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SBUS<n>:CAN:TRIGger:PATtern:DATA"](#) on page 1019
 - [":SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth"](#) on page 1021
 - [":SBUS<n>:CAN:TRIGger:PATtern:ID"](#) on page 1023
 - [":SBUS<n>:CAN:TRIGger:PATtern:ID:MODE"](#) on page 1024
 - [":SBUS<n>:CAN:TRIGger:IDFilter"](#) on page 1018
 - [":SBUS<n>:CAN:SIGNal:DEFinition"](#) on page 1011
 - [":SBUS<n>:CAN:SOURce"](#) on page 1014
 - [":RECall:DBC\[:START\]"](#) on page 945
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:MESSage"](#) on page 1025
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:SIGNal"](#) on page 1026
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:VALue"](#) on page 1027

:SBUS<n>:CAN:TRIGger:IDFilter

N (see [page 1708](#))

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 1015

:SBUS<n>:CAN:TRIGger:PATtern:DATA

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth"](#) on page 1021
 • [":SBUS<n>:CAN:TRIGger:PATtern:ID"](#) on page 1023

:SBUS<n>:CAN:TRIGger:PATtern:DATA:DLC

N (see [page 1708](#))

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:DLC?

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 1019

:SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:CAN:TRIGger:PATtern:DATA"](#) on page 1019
 • [":SBUS<n>:CAN:SOURce"](#) on page 1014

:SBUS<n>:CAN:TRIGger:PATtern:DATA:START

N (see [page 1708](#))

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 1019
 - [":SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth"](#) on page 1021

:SBUS<n>:CAN:TRIGger:PATtern:ID

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:CAN:TRIGger:PATtern:ID:MODE"](#) on page 1024
 • [":SBUS<n>:CAN:TRIGger:PATtern:DATA"](#) on page 1019

:SBUS<n>:CAN:TRIGger:PATtern:ID:MODE

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479

- [":SBUS<n>:MODE"](#) on page 987
- [":SBUS<n>:CAN:TRIGger:PATtern:DATA"](#) on page 1019
- [":SBUS<n>:CAN:TRIGger:PATtern:DATA:LENGth"](#) on page 1021
- [":SBUS<n>:CAN:TRIGger:PATtern:ID"](#) on page 1023

:SBUS<n>:CAN:TRIGger:SYMBolic:MESSage

N (see [page 1708](#))

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 945
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:SIGNal"](#) on page 1026
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:VALue"](#) on page 1027

:SBUS<n>:CAN:TRIGger:SYMBolic:SIGNaL

N (see [page 1708](#))

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 945
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:MESSage"](#) on page 1025
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:VALue"](#) on page 1027

:SBUS<n>:CAN:TRIGger:SYMBolic:VALue

N (see [page 1708](#))

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 945
 - [":SBUS<n>:CAN:TRIGger"](#) on page 1015
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:MESSage"](#) on page 1025
 - [":SBUS<n>:CAN:TRIGger:SYMBolic:SIGNaL"](#) on page 1026

:SBUS<n>:CAN:TRIGger:TYPE

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN:TRIGger:TYPE <type>

<type> ::= {STANdard | FD | XL}

The :SBUS<n>:CAN:TRIGger:TYPE command selects the CAN trigger type:

- STANdard – CAN Standard trigger, available with all CAN decode types.
- FD – CAN Flexible Data trigger, available with the FD (Flexible Data), XFAST (CAN XL Fast Mode), or XSIC (CAN XL SIC Mode) decode types.
- XL – CAN XL trigger, available with the XFAST (CAN XL Fast Mode) or XSIC (CAN XL SIC Mode) decode types.

The CAN decode type is selected with the :SBUS<n>:CAN:TYPE command.

Query Syntax :SBUS<n>:CAN:TRIGger:TYPE?

The :SBUS<n>:CAN:TRIGger:TYPE? query returns the selected CAN trigger type.

Return Format <type><NL>

<type> ::= {STAN | FD | XL}

- See Also**
- [":SBUS<n>:CAN:TYPE"](#) on page 1029
 - [":SBUS<n>:CAN:SIGNAL:BAUDrate"](#) on page 1010
 - [":SBUS<n>:CAN:SIGNAL:FDBaudrate"](#) on page 1012
 - [":SBUS<n>:CAN:SIGNAL:XLBaudrate"](#) on page 1013
 - [":SBUS<n>:CAN:SAMPLEpoint"](#) on page 1009
 - [":SBUS<n>:CAN:FDSPoint"](#) on page 1008
 - [":SBUS<n>:CAN:XLSPoint"](#) on page 1030

:SBUS<n>:CAN:TYPE

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN:TYPE <type>

<type> ::= {STANdard | FD | XFAST | XSIC}

The :SBUS<n>:CAN:TYPE command selects the CAN protocol decode type:

- STANdard – CAN Standard decode (data field length from 1 to 8 bytes).
- FD – CAN Flexible Data decode (data field length from 1 to 64 bytes).
- XFAST – CAN XL (Fast Mode) decode (arbitration mode and data mode, data field length from 1 to 2048 bytes). Fast mode is active when there is PWM encoding and decoding is happening in OSI layers.
- XSIC – CAN XL (SIC Mode) decode (slow mode, arbitration mode, same speeds as CAN-FD).

Query Syntax :SBUS<n>:CAN:TYPE?

The :SBUS<n>:CAN:TYPE? query returns the selected CAN protocol decode type.

Return Format <type><NL>

<type> ::= {STAN | FD | XFAS | XSIC}

- See Also**
- [":SBUS<n>:CAN:SIGNAL:BAUDrate"](#) on page 1010
 - [":SBUS<n>:CAN:SIGNAL:FDBaudrate"](#) on page 1012
 - [":SBUS<n>:CAN:SIGNAL:XLBaudrate"](#) on page 1013
 - [":SBUS<n>:CAN:SAMPLEpoint"](#) on page 1009
 - [":SBUS<n>:CAN:FDSPoint"](#) on page 1008
 - [":SBUS<n>:CAN:XLSPoint"](#) on page 1030

:SBUS<n>:CAN:XLSPoint

N (see [page 1708](#))

Command Syntax :SBUS<n>:CAN:XLSPoint <value>

<value> ::= even numbered percentages from 30 to 90 in NR3 format.

The :SBUS<n>:CAN:XLSPoint 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:XLSPoint?

The :SBUS<n>:CAN:XLSPoint? query returns the current CAN XL sample point setting.

Return Format <value><NL>

<value> ::= even numbered percentages from 30 to 90 in NR3 format.

- See Also**
- [":SBUS<n>:CAN:SIGNal:XLBaudrate"](#) on page 1013
 - [":SBUS<n>:CAN:TYPE"](#) on page 1029

:SBUS<n>:IIC Commands

NOTE

These commands are valid when the low-speed IIC and SPI serial decode option has been licensed.

Table 120 :SBUS<n>:IIC Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:IIC:ASize <size> (see page 1032)	:SBUS<n>:IIC:ASize? (see page 1032)	<size> ::= {BIT7 BIT8}
:SBUS<n>:IIC[:SOURce] :CLOCK <source> (see page 1033)	:SBUS<n>:IIC[:SOURce] :CLOCK? (see page 1033)	<source> ::= {CHANnel<n> DIGital<d> } <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 1034)	:SBUS<n>:IIC[:SOURce] :DATA? (see page 1034)	<source> ::= {CHANnel<n> DIGital<d> } <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 1035)	:SBUS<n>:IIC:TRIGger: PATtern:ADDRESS? (see page 1035)	<value> ::= integer
:SBUS<n>:IIC:TRIGger: PATtern:DATA <value> (see page 1036)	:SBUS<n>:IIC:TRIGger: PATtern:DATA? (see page 1036)	<value> ::= integer
:SBUS<n>:IIC:TRIGger: PATtern:DATA2 <value> (see page 1037)	:SBUS<n>:IIC:TRIGger: PATtern:DATA2? (see page 1037)	<value> ::= integer
:SBUS<n>:IIC:TRIGger: QUALifier <value> (see page 1038)	:SBUS<n>:IIC:TRIGger: QUALifier? (see page 1038)	<value> ::= {EQUAL NOTequal LESSthan GREATERthan}
:SBUS<n>:IIC:TRIGger[:TYPE] <type> (see page 1039)	:SBUS<n>:IIC:TRIGger[:TYPE]? (see page 1039)	<type> ::= {START STOP REStart ADDRESS ANACK DNACK NACKnowledge REPRom READ7 WRITe7 R7Data2 W7Data2 WRITe10}

:SBUS<n>:IIC:ASIZe

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983

• [":SBUS<n>:IIC Commands"](#) on page 1031

:SBUS<n>:IIC[:SOURce]:CLOCK

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC[:SOURce]:CLOCK <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>: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 1479
 - [":SBUS<n>:IIC\[:SOURce\]:DATA"](#) on page 1034

:SBUS<n>:IIC[:SOURce]:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC[:SOURce]:DATA <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>: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 1479
 - [":SBUS<n>:IIC\[:SOURce\]:CLOCK"](#) on page 1033

:SBUS<n>:IIC:TRIGger:PATtern:ADDRes

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC:TRIGger:PATtern:ADDRes <value>

<value> ::= integer

The :SBUS<n>:IIC:TRIGger:PATtern:ADDRes command sets the address for IIC data. The address can range from 0 to 127 (7-bit) or 1023 (10-bit). Use the don't care address (-1) to ignore the address value.

Query Syntax :SBUS<n>:IIC:TRIGger:PATtern:ADDRes?

The :SBUS<n>:IIC:TRIGger:PATtern:ADDRes? query returns the current address for IIC data.

Return Format <value><NL>

<value> ::= integer

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:IIC:TRIGger:PATtern:DATA"](#) on page 1036
 - [":SBUS<n>:IIC:TRIGger:PATtern:DATA2"](#) on page 1037
 - [":SBUS<n>:IIC:TRIGger\[:TYPE\]"](#) on page 1039

:SBUS<n>:IIC:TRIGger:PATtern:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC:TRIGger:PATtern:DATA <value>

<value> ::= integer

The :SBUS<n>:IIC:TRIGger:PATtern:DATA command sets IIC data. The data value can range from 0 to 255. Use the don't care data pattern (-1) 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 1479
 - [":SBUS<n>:IIC:TRIGger:PATtern:ADDRESS"](#) on page 1035
 - [":SBUS<n>:IIC:TRIGger:PATtern:DATA2"](#) on page 1037
 - [":SBUS<n>:IIC:TRIGger\[:TYPE\]"](#) on page 1039
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:SBUS<n>:IIC:TRIGger:PATtern:DATa2

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC:TRIGger:PATtern:DATa2 <value>

<value> ::= integer

The :SBUS<n>:IIC:TRIGger:PATtern:DATa2 command sets IIC data 2. The data value can range from 0 to 255. Use the don't care data pattern (-1) 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 1479
 - [":SBUS<n>:IIC:TRIGger:PATtern:ADDReSS"](#) on page 1035
 - [":SBUS<n>:IIC:TRIGger:PATtern:DATA"](#) on page 1036
 - [":SBUS<n>:IIC:TRIGger\[:TYPE\]"](#) on page 1039
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:SBUS<n>:IIC:TRIGger:QUALifier

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC:TRIGger:QUALifier <value>

<value> ::= {EQUAL | NOTequal | LESSthan | GREATERthan}

The :SBUS<n>:IIC:TRIGger:QUALifier command sets the IIC data qualifier when TRIGger:IIC:TRIGger[:TYPE] is set to REPRom.

Query Syntax :SBUS<n>:IIC:TRIGger:QUALifier?

The :SBUS<n>:IIC:TRIGger:QUALifier? query returns the current IIC data qualifier value.

Return Format <value><NL>

<value> ::= {EQUAL | NOTequal | LESSthan | GREATERthan}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:IIC:TRIGger\[:TYPE\]"](#) on page 1039

:SBUS<n>:IIC:TRIGger[:TYPE]

N (see [page 1708](#))

Command Syntax :SBUS<n>:IIC:TRIGger[:TYPE] <value>

```
<value> ::= {START | STOP | REStart | ADDRess | ANACk | DNACk
             | NACKnowledge | REPRom | READ7 | WRITe7 | R7Data2 | W7Data2
             | WRITe10}
```

The :SBUS<n>:IIC:TRIGger[:TYPE] command sets the IIC trigger type:

- START – Start condition.
- STOP – Stop condition.
- REStart – Another start condition occurs before a stop condition.
- ADDRess – Triggers on the selected address. The R/W bit is ignored.
- ANACk – Address with no acknowledge.
- DNACk – Write data with no acknowledge.
- NACKnowledge – Missing acknowledge.
- REPRom – EEPROM data read.
- READ7 – 7-bit address frame containing (Start:Address7:Read:Ack:Data). The value READ is also accepted for READ7.
- WRITe7 – 7-bit address frame containing (Start:Address7:Write:Ack:Data). The value WRITe is also accepted for WRITe7.
- R7Data2 – 7-bit address frame containing (Start:Address7:Read:Ack:Data:Ack:Data2).
- 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).

NOTE

The short form of READ7 (READ7), REPRom (REPR), WRITe7 (WRIT7), and WRITe10 (WRIT10) do not follow the defined Long Form to Short Form Truncation Rules (see [page 1710](#)).

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 | REST | ADDR | ANAC | DNAC | NACK | REPR
             | READ7 | WRIT7 | R7D2 | W7D2 | WRIT10}
```

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494

- [":SBUS<n>:IIC:TRIGger:PATtern:ADDReSS"](#) on page 1035
- [":SBUS<n>:IIC:TRIGger:PATtern:DATA"](#) on page 1036
- [":SBUS<n>:IIC:TRIGger:PATtern:DATa2"](#) on page 1037
- [":SBUS<n>:IIC:TRIGger:QUALifier"](#) on page 1038
- ["Long Form to Short Form Truncation Rules"](#) on page 1710

:SBUS<n>:LIN Commands

NOTE

These commands are valid when the Automotive Serial Analysis license has been enabled.

NOTE

Aerospace protocol decodes (ARINC 429 or MIL-STD-155) and Automotive protocol decodes (CAN, LIN, or SENT) cannot be used at the same time. Protocol decode mode choices on Protocol2 are limited by the last Aerospace or Automotive protocol decode mode selected on Protocol1.

Table 121 :SBUS<n>:LIN Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:LIN:DISPlay <type> (see page 1043)	:SBUS<n>:LIN:DISPlay? (see page 1043)	<type> ::= {HEXadecimal SYMBolic}
:SBUS<n>:LIN:PARity { {0 OFF} {1 ON} } (see page 1044)	:SBUS<n>:LIN:PARity? (see page 1044)	{0 1}
:SBUS<n>:LIN:SAMPlEpo int <value> (see page 1045)	:SBUS<n>:LIN:SAMPlEpo int? (see page 1045)	<value> ::= {60 62.5 68 70 75 80 87.5} in NR3 format
:SBUS<n>:LIN:SIGNAL:B AUDrate <baudrate> (see page 1046)	:SBUS<n>:LIN:SIGNAL:B AUDrate? (see page 1046)	<baudrate> ::= integer from 2400 to 625000 in 100 b/s increments
:SBUS<n>:LIN:SOURce <source> (see page 1047)	:SBUS<n>:LIN:SOURce? (see page 1047)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:LIN:STANDard <std> (see page 1048)	:SBUS<n>:LIN:STANDard ? (see page 1048)	<std> ::= {LIN13 LIN13NLC LIN20}
:SBUS<n>:LIN:SYNCbrea k <value> (see page 1049)	:SBUS<n>:LIN:SYNCbrea k? (see page 1049)	<value> ::= integer = {11 12 13}
:SBUS<n>:LIN:TRIGger <condition> (see page 1050)	:SBUS<n>:LIN:TRIGger? (see page 1050)	<condition> ::= {SYNCbreak ID DATA}

Table 121 :SBUS<n>:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:LIN:TRIGger:ID <value> (see page 1051)	:SBUS<n>:LIN:TRIGger:ID? (see page 1051)	<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 1052)	:SBUS<n>:LIN:TRIGger:PATtern:DATA? (see page 1052)	<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 1054)	:SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth? (see page 1054)	<length> ::= integer from 1 to 8 in NR1 format
:SBUS<n>:LIN:TRIGger:PATtern:FORMat <base> (see page 1055)	:SBUS<n>:LIN:TRIGger:PATtern:FORMat? (see page 1055)	<base> ::= {BINary HEX DECimal}
:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe <name> (see page 1056)	:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe? (see page 1056)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal <name> (see page 1057)	:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal? (see page 1057)	<name> ::= quoted ASCII string
:SBUS<n>:LIN:TRIGger:SYMBolic:VALue <data> (see page 1058)	:SBUS<n>:LIN:TRIGger:SYMBolic:VALue? (see page 1058)	<data> ::= value in NR3 format

:SBUS<n>:LIN:DISPlay

N (see [page 1708](#))

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 947

:SBUS<n>:LIN:PARity

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983

• [":SBUS<n>:LIN Commands"](#) on page 1041

:SBUS<n>:LIN:SAMPlendpoint

N (see [page 1708](#))

Command Syntax :SBUS<n>:LIN:SAMPlendpoint <value>

<value><NL>

<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format

The :SBUS<n>:LIN:SAMPlendpoint 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:SAMPlendpoint?

The :SBUS<n>:LIN:SAMPlendpoint? 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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050

:SBUS<n>:LIN:SIGNal:BAUDrate

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:SOURce"](#) on page 1047

:SBUS<n>:LIN:SOURce

N (see [page 1708](#))

Command Syntax :SBUS<n>:LIN: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>: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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050

:SBUS<n>:LIN:STANdard

N (see [page 1708](#))

Command Syntax :SBUS<n>:LIN:STANdard <std>

<std> ::= {LIN13 | LIN13NLC | LIN20}

The :SBUS<n>:LIN:STANdard command sets the LIN standard in effect for triggering and decoding:

- LIN13 – LIN 1.3.
- LIN13NLC – LIN 1.3 (no length control). Select this for systems where length control is not used and all nodes have knowledge of the data packet size. In LIN 1.3, the ID may or may not be used to indicate the number of bytes. (In LIN 2.X, there is no length control.)
- LIN20 – LIN 2.X.

For LIN 1.2 signals, use the LIN 1.3 setting. The LIN 1.3 setting assumes the signal follows the "Table of Valid ID Values" as shown in section A.2 of the LIN Specification dated December 12, 2002. If your signal does not comply with the table, use the LIN 2.X setting.

Query Syntax :SBUS<n>:LIN:STANdard?

The :SBUS<n>:LIN:STANdard? query returns the current LIN standard setting.

Return Format <std><NL>

<std> ::= {LIN13 | LIN13NLC | LIN20}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:SOURce"](#) on page 1047

:SBUS<n>:LIN:SYNCbreak

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:SOURce"](#) on page 1047

:SBUS<n>:LIN:TRIGger

N (see [page 1708](#))

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 1663

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:LIN:TRIGger:ID"](#) on page 1051
 - [":SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth"](#) on page 1054
 - [":SBUS<n>:LIN:TRIGger:PATtern:DATA"](#) on page 1052
 - [":SBUS<n>:LIN:SOURce"](#) on page 1047
 - [":RECall:LDF\[:START\]"](#) on page 947
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:FRAME"](#) on page 1056
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:SIGNAL"](#) on page 1057
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:VALue"](#) on page 1058

:SBUS<n>:LIN:TRIGger:ID

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
• [":TRIGger:MODE"](#) on page 1494
• [":SBUS<n>:LIN:TRIGger"](#) on page 1050
• [":SBUS<n>:LIN:SOURce"](#) on page 1047

:SBUS<n>:LIN:TRIGger:PATtern:DATA

N (see [page 1708](#))

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> ::= "0xn...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 1479
 - [":SBUS<n>:LIN:TRIGger:PATtern:FORMat"](#) on page 1055
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth"](#) on page 1054

:SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:LIN:TRIGger:PATtern:DATA"](#) on page 1052
 • [":SBUS<n>:LIN:SOURce"](#) on page 1047

:SBUS<n>:LIN:TRIGger:PATtern:FORMat

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:LIN:TRIGger:PATtern:DATA"](#) on page 1052
 - [":SBUS<n>:LIN:TRIGger:PATtern:DATA:LENGth"](#) on page 1054

:SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe

N (see [page 1708](#))

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 947
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:SIGNaL"](#) on page 1057
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:VALue"](#) on page 1058

:SBUS<n>:LIN:TRIGger:SYMBolic:SIGNal

N (see [page 1708](#))

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 947
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:FRAMe"](#) on page 1056
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:VALue"](#) on page 1058

:SBUS<n>:LIN:TRIGger:SYMBolic:VALue

N (see [page 1708](#))

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 947
 - [":SBUS<n>:LIN:TRIGger"](#) on page 1050
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:FRAME"](#) on page 1056
 - [":SBUS<n>:LIN:TRIGger:SYMBolic:SIGNAL"](#) on page 1057

:SBUS<n>:M1553 Commands

NOTE

These commands are valid when the Aerospace and Defense Serial Analysis license has been enabled.

NOTE

Aerospace protocol decodes (ARINC 429 or MIL-STD-155) and Automotive protocol decodes (CAN, LIN, or SENT) cannot be used at the same time. Protocol decode mode choices on Protocol2 are limited by the last Aerospace or Automotive protocol decode mode selected on Protocol1.

Table 122 :SBUS<n>:M1553 Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:M1553:AUTose tup (see page 1060)	n/a	n/a
:SBUS<n>:M1553:BASE <base> (see page 1061)	:SBUS<n>:M1553:BASE? (see page 1061)	<base> ::= {BINary HEX}
:SBUS<n>:M1553:SOURce <source> (see page 1062)	:SBUS<n>:M1553:SOURce ? (see page 1062)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:M1553:TRIGge r:PATtern:DATA <string> (see page 1063)	:SBUS<n>:M1553:TRIGge r:PATtern:DATA? (see page 1063)	<string> ::= "nn...n" where n ::= {0 1 X}
:SBUS<n>:M1553:TRIGge r:RTA <value> (see page 1064)	:SBUS<n>:M1553:TRIGge r:RTA? (see page 1064)	<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 1065)	:SBUS<n>:M1553:TRIGge r:TYPE? (see page 1065)	<type> ::= {DStart DSTOp CStart CSTOp RTA PERRor SERRor MERRor RTA11}

:SBUS<n>:M1553:AUTosetup

N (see [page 1708](#))

Command Syntax :SBUS<n>:M1553: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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:M1553:SOURce"](#) on page 1062

:SBUS<n>:M1553:BASE

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
 • [":SBUS<n>:M1553 Commands"](#) on page 1059

:SBUS<n>:M1553:SOURce

N (see [page 1708](#))

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 1492
 - [":TRIGger:LEVel:LOW"](#) on page 1493
 - [":TRIGger:MODE"](#) on page 1494
 - ["Introduction to :TRIGger Commands"](#) on page 1479

:SBUS<n>:M1553:TRIGger:PATtern:DATA

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:M1553:TRIGger:TYPE"](#) on page 1065
 - [":SBUS<n>:M1553:TRIGger:RTA"](#) on page 1064

:SBUS<n>:M1553:TRIGger:RTA

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:M1553:TRIGger:TYPE"](#) on page 1065

:SBUS<n>:M1553:TRIGger:TYPE

N (see [page 1708](#))

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> ::= {DST | DSTO | CST | CSTO | RTA | PERR | SERR
| MERR | RTA11}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:M1553:TRIGger:RTA"](#) on page 1064
 - [":SBUS<n>:M1553:TRIGger:PATtern:DATA"](#) on page 1063
 - [":TRIGger:MODE"](#) on page 1494

:SBUS<n>:SENT Commands

NOTE

These commands are valid when the Automotive Serial Analysis license has been enabled.

NOTE

Aerospace protocol decodes (ARINC 429 or MIL-STD-155) and Automotive protocol decodes (CAN, LIN, or SENT) cannot be used at the same time. Protocol decode mode choices on Protocol2 are limited by the last Aerospace or Automotive protocol decode mode selected on Protocol1.

Table 123 :SBUS<n>:SENT Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SENT:CLOCK <period> (see page 1069)	:SBUS<n>:SENT:CLOCK? (see page 1069)	<period> ::= the nominal clock period (tick), from 500 ns to 300 us, in NR3 format.
:SBUS<n>:SENT:CRC <format> (see page 1070)	:SBUS<n>:SENT:CRC? (see page 1070)	<format> ::= {LEGacy RECommended}
:SBUS<n>:SENT:DISPlay <base> (see page 1071)	:SBUS<n>:SENT:DISPlay ? (see page 1071)	<base> ::= {HEX DECimal SYMBolic}
:SBUS<n>:SENT:FORMat <decode> (see page 1073)	:SBUS<n>:SENT:FORMat? (see page 1073)	<decode> ::= {NIBBles FSIGnal FSSerial FESerial SSERial ESERial}
:SBUS<n>:SENT:IDLE <state> (see page 1075)	:SBUS<n>:SENT:IDLE? (see page 1075)	<state> ::= {LOW HIGH}
:SBUS<n>:SENT:LENGth <#_nibbles> (see page 1076)	:SBUS<n>:SENT:LENGth? (see page 1076)	<#_nibbles> ::= from 1-6, in NR1 format.
:SBUS<n>:SENT:PPULse { {0 OFF} {1 ON} SPC } (see page 1077)	:SBUS<n>:SENT:PPULse? (see page 1077)	{ 0 1 SPC }
:SBUS<n>:SENT:SIGNaL<s>:DISPlay { {0 OFF} {1 ON} } (see page 1079)	:SBUS<n>:SENT:SIGNaL<s>:DISPlay? (see page 1079)	<s> ::= 1-6, in NR1 format. { 0 1 }
:SBUS<n>:SENT:SIGNaL<s>:LENGth <length> (see page 1080)	:SBUS<n>:SENT:SIGNaL<s>:LENGth? (see page 1080)	<s> ::= 1-6, in NR1 format. <length> ::= from 1-24, in NR1 format.

Table 123 :SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:SIGNAl<s>:MULTiplier<multiplier> (see page 1082)	:SBUS<n>:SENT:SIGNAl<s>:MULTiplier? (see page 1082)	<s> ::= 1-6, in NR1 format. <multiplier> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNAl<s>:OFFSet <offset> (see page 1084)	:SBUS<n>:SENT:SIGNAl<s>:OFFSet? (see page 1084)	<s> ::= 1-6, in NR1 format. <offset> ::= from 1-24, in NR3 format.
:SBUS<n>:SENT:SIGNAl<s>:ORDER <order> (see page 1086)	:SBUS<n>:SENT:SIGNAl<s>:ORDER? (see page 1086)	<s> ::= 1-6, in NR1 format. <order> ::= {MSNFirst LSNFirst}
:SBUS<n>:SENT:SIGNAl<s>:START <position> (see page 1088)	:SBUS<n>:SENT:SIGNAl<s>:START? (see page 1088)	<s> ::= 1-6, in NR1 format. <position> ::= from 0-23, in NR1 format.
:SBUS<n>:SENT:SOURce<source> (see page 1090)	:SBUS<n>:SENT:SOURce? (see page 1090)	<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 page 1092)	:SBUS<n>:SENT:TOLerance? (see page 1092)	<percent> ::= from 3-30, in NR1 format.
:SBUS<n>:SENT:TRIGger<mode> (see page 1093)	:SBUS<n>:SENT:TRIGger? (see page 1093)	<mode> ::= {SFCMessage SSCMessage FCData SCMid SCData FCCerror SCCerror CRCerror TOLerror PPErrror SSPerror}
:SBUS<n>:SENT:TRIGger:FAST:DATA <string> (see page 1095)	:SBUS<n>:SENT:TRIGger:FAST:DATA? (see page 1095)	<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 1096)	:SBUS<n>:SENT:TRIGger:SLOW:DATA? (see page 1096)	<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.

Table 123 :SBUS<n>:SENT Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SENT:TRIGger :SLOW:ID <id> (see page 1098)	:SBUS<n>:SENT:TRIGger :SLOW:ID? (see page 1098)	<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:ILENgth <length> (see page 1100)	:SBUS<n>:SENT:TRIGger :SLOW:ILENgth? (see page 1100)	<length> ::= {SHORt LONG}
:SBUS<n>:SENT:TRIGger :TOLerance <percent> (see page 1101)	:SBUS<n>:SENT:TRIGger :TOLerance? (see page 1101)	<percent> ::= from 1-18, in NR1 format.

:SBUS<n>:SENT:CLOCK

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:CLOCK <period>

<period> ::= the nominal clock period (tick), from 500 ns to 300 us, in NR3 format.

The :SBUS<n>:SENT:CLOCK command specifies the nominal clock period (tick), from 500 ns to 300 μs.

Query Syntax :SBUS<n>:SENT:CLOCK?

The :SBUS<n>:SENT:CLOCK? query returns the clock period setting.

Return Format <period><NL>

<period> ::= the nominal clock period (tick), from 500 ns to 300 us, in NR3 format.

- See Also**
- [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENGth"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:CRC

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:STARt"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:DISPlay

N (see [page 1708](#))

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:

$$\text{PhysicalValue} = (\text{Multiplier} * \text{SignalValueAsUnsignedInteger}) + \text{Offset}$$

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095

- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:FORMat

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:FORMat <decode>

<decode> ::= {NIBBles | FSIGnal | FSSerial | FESerial | SSERial | ESERial}

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.
- ESERial – 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:SIGNa<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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075

- **":SBUS<n>:SENT:LENGth"** on page 1076
- **":SBUS<n>:SENT:PPULse"** on page 1077
- **":SBUS<n>:SENT:SIGNAl<s>:DISPlay"** on page 1079
- **":SBUS<n>:SENT:SIGNAl<s>:LENGth"** on page 1080
- **":SBUS<n>:SENT:SIGNAl<s>:MULTIplier"** on page 1082
- **":SBUS<n>:SENT:SIGNAl<s>:OFFSet"** on page 1084
- **":SBUS<n>:SENT:SIGNAl<s>:ORDer"** on page 1086
- **":SBUS<n>:SENT:SIGNAl<s>:STARt"** on page 1088
- **":SBUS<n>:SENT:SOURce"** on page 1090
- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger"** on page 1093
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:IDLE

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:LENGth

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:PPULSe

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:PPULSe {{0 | OFF} | {1 | ON} | SPC}

The :SBUS<n>:SENT:PPULSe command specifies whether there is a pause pulse between Fast Channel Messages:

- OFF – There is no pause pulse between Fast Channel Messages.

Note that a SENT serial bus with no pause pulse is never idle. This means, during normal operation, the fast channel decode line will show a continuous stream of packets, with a new packet opening as soon as the previous one has closed.

- ON – Pause pulses are added between Fast Channel Messages so that frames come at a regular interval.

If there is a pause pulse (and **Pause Pulse** is on), idle time is shown between messages.

- SPC (Short PWM Code) – In SENT SPC, there are no pause pulses. Instead, the message event is triggered by the master when it wants to receive data. SENT SPC ends the transmission after the CRC so it almost appears as if there is a pause pulse from the end until the next master trigger.

Query Syntax :SBUS<n>:SENT:PPULSe?

The :SBUS<n>:SENT:PPULSe? query returns the pause mode setting.

Return Format <setting><NL>

<setting> ::= {0 | 1 | SPC}

- See Also**
- [":SBUS<n>:SENT:CLOCK"](#) on page 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090

- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger"** on page 1093
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SIGNAl<s>:DISPlay

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAl<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAl<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNAl<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNAl<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAl<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgth"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:SIGNal<s>:LENGth

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:SIGNal<s>:LENGth <length>

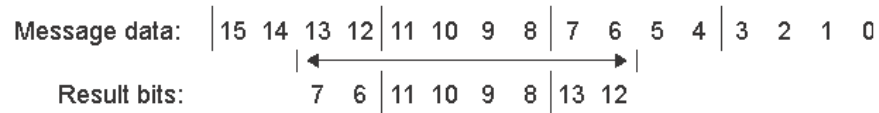
<s> ::= 1-6, in NR1 format.

<length> ::= from 1-24, in NR1 format.

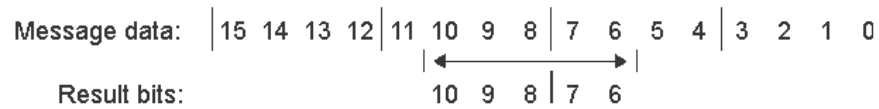
The :SBUS<n>:SENT:SIGNal<s>:LENGth command specifies the bit length of the signal being defined.

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084

- **":SBUS<n>:SENT:SIGNAl<s>:ORDer"** on page 1086
- **":SBUS<n>:SENT:SIGNAl<s>:START"** on page 1088
- **":SBUS<n>:SENT:SOURce"** on page 1090
- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger"** on page 1093
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SIGNAl<s>:MULTIplier

N (see [page 1708](#))

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.

- $\text{PhysicalValue} = (\text{Multiplier} * \text{SignalValueAsUnsignedInteger}) + \text{Offset}$

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAl<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNAl<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAl<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNAl<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAl<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100

- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SIGNAL<s>:OFFSet

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAL<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNAL<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAL<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNAL<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAL<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100

- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SIGNal<s>:ORDER

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:SIGNal<s>:ORDER <order>

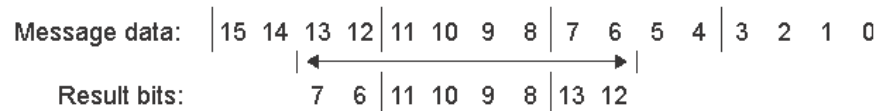
<s> ::= 1-6, in NR1 format.

<order> ::= {MSNFirst | LSNFirst}

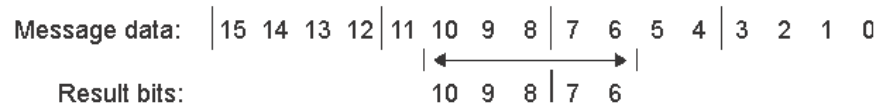
The :SBUS<n>:SENT:SIGNal<s>:ORDER command specifies the nibble order of the signal being defined, either Most Significant Nibble first, or Least Significant Nibble first.

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082

- **":SBUS<n>:SENT:SIGNAl<s>:OFFSet"** on page 1084
- **":SBUS<n>:SENT:SIGNAl<s>:START"** on page 1088
- **":SBUS<n>:SENT:SOURce"** on page 1090
- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger"** on page 1093
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SIGNal<s>:START

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:SIGNal<s>:START <position>

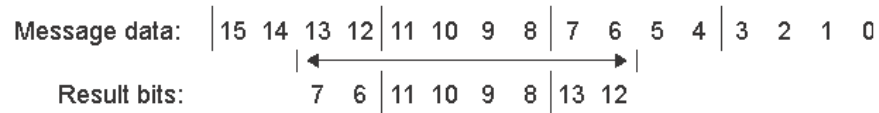
<s> ::= 1-6, in NR1 format.

<position> ::= from 0-23, in NR1 format.

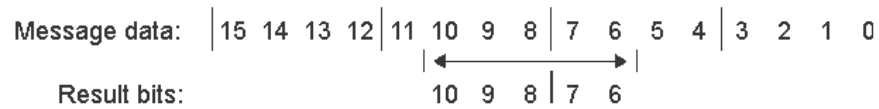
The :SBUS<n>:SENT:SIGNal<s>:START command specifies the starting bit of the Fast Signal being defined.

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082

- **":SBUS<n>:SENT:SIGNAl<s>:OFFSet"** on page 1084
- **":SBUS<n>:SENT:SIGNAl<s>:ORDer"** on page 1086
- **":SBUS<n>:SENT:SOURce"** on page 1090
- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger"** on page 1093
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:SOURce

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAL<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNAL<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAL<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNAL<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNAL<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAL<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgth"](#) on page 1100

- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:TOLerance

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:TRIGger

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:TRIGger <mode>

```
<mode> ::= {SFCMessage | SSCMessage | FCData | 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.
- FCData – triggers on a Fast Channel Message when the Status & Communication nibble and the data nibbles match the values entered.
- SCMid – triggers when a Slow Channel Message ID matches the value entered.
- SCData – triggers when a Slow Channel Message ID and Data field both match the values entered.
- 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 1069
 - **"[:SBUS<n>:SENT:CRC](#)"** on page 1070
 - **"[:SBUS<n>:SENT:DISPlay](#)"** on page 1071
 - **"[:SBUS<n>:SENT:FORMat](#)"** on page 1073
 - **"[:SBUS<n>:SENT:IDLE](#)"** on page 1075
 - **"[:SBUS<n>:SENT:LENGth](#)"** on page 1076

- **":SBUS<n>:SENT:PPULse"** on page 1077
- **":SBUS<n>:SENT:SIGNAl<s>:DISPlay"** on page 1079
- **":SBUS<n>:SENT:SIGNAl<s>:LENGth"** on page 1080
- **":SBUS<n>:SENT:SIGNAl<s>:MULTiplier"** on page 1082
- **":SBUS<n>:SENT:SIGNAl<s>:OFFSet"** on page 1084
- **":SBUS<n>:SENT:SIGNAl<s>:ORDer"** on page 1086
- **":SBUS<n>:SENT:SIGNAl<s>:START"** on page 1088
- **":SBUS<n>:SENT:SOURce"** on page 1090
- **":SBUS<n>:SENT:TOLerance"** on page 1092
- **":SBUS<n>:SENT:TRIGger:FAST:DATA"** on page 1095
- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ID"** on page 1098
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:TRIGger:FAST:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:TRIGger:FAST:DATA <string>

<string> ::= "nnnn..." where n ::= {0 | 1 | X}

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

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.

Query Syntax :SBUS<n>:SENT:TRIGger:FAST:DATA?

The :SBUS<n>:SENT:TRIGger:FAST:DATA? query returns the fast channel data trigger setting.

Return Format <string><NL>

<string> ::= "nnnn..." where n ::= {0 | 1 | X}

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

- See Also**
- [":SBUS<n>:SENT:CLOCK"](#) on page 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAL<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNAL<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAL<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNAL<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNAL<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAL<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgth"](#) on page 1100
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:TRIGger:SLOW:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:SENT:TRIGger:SLOW:DATA <data>

<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.

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 format.

<data> ::= when ILENgth = LONG, from -1 (don't care) to 4095, in NR1 format.

- See Also**
- [":SBUS<n>:SENT:CLOCK"](#) on page 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENgth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENgth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:STARt"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098

- `":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"` on page 1100
- `":SBUS<n>:SENT:TRIGger:TOLerance"` on page 1101

:SBUS<n>:SENT:TRIGger:SLOW:ID

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENgth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENgth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTiplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:STARt"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095

- **":SBUS<n>:SENT:TRIGger:SLOW:DATA"** on page 1096
- **":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"** on page 1100
- **":SBUS<n>:SENT:TRIGger:TOLerance"** on page 1101

:SBUS<n>:SENT:TRIGger:SLOW:ILENgtH

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMAt"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNal<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNal<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNal<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNal<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNal<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNal<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:TOLerance"](#) on page 1101

:SBUS<n>:SENT:TRIGger:TOLerance

N (see [page 1708](#))

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 1069
 - [":SBUS<n>:SENT:CRC"](#) on page 1070
 - [":SBUS<n>:SENT:DISPlay"](#) on page 1071
 - [":SBUS<n>:SENT:FORMat"](#) on page 1073
 - [":SBUS<n>:SENT:IDLE"](#) on page 1075
 - [":SBUS<n>:SENT:LENGth"](#) on page 1076
 - [":SBUS<n>:SENT:PPULse"](#) on page 1077
 - [":SBUS<n>:SENT:SIGNAL<s>:DISPlay"](#) on page 1079
 - [":SBUS<n>:SENT:SIGNAL<s>:LENGth"](#) on page 1080
 - [":SBUS<n>:SENT:SIGNAL<s>:MULTIplier"](#) on page 1082
 - [":SBUS<n>:SENT:SIGNAL<s>:OFFSet"](#) on page 1084
 - [":SBUS<n>:SENT:SIGNAL<s>:ORDer"](#) on page 1086
 - [":SBUS<n>:SENT:SIGNAL<s>:START"](#) on page 1088
 - [":SBUS<n>:SENT:SOURce"](#) on page 1090
 - [":SBUS<n>:SENT:TOLerance"](#) on page 1092
 - [":SBUS<n>:SENT:TRIGger"](#) on page 1093
 - [":SBUS<n>:SENT:TRIGger:FAST:DATA"](#) on page 1095
 - [":SBUS<n>:SENT:TRIGger:SLOW:DATA"](#) on page 1096
 - [":SBUS<n>:SENT:TRIGger:SLOW:ID"](#) on page 1098
 - [":SBUS<n>:SENT:TRIGger:SLOW:ILENgtH"](#) on page 1100

:SBUS<n>:SPI Commands

NOTE

These commands are only valid when the low-speed IIC and SPI serial decode option has been licensed.

Table 124 :SBUS<n>:SPI Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:SPI:BITOrder<order> (see page 1104)	:SBUS<n>:SPI:BITOrder? (see page 1104)	<order> ::= {LSBFirst MSBFirst}
:SBUS<n>:SPI:CLOCK:SLOPe <slope> (see page 1105)	:SBUS<n>:SPI:CLOCK:SLOPe? (see page 1105)	<slope> ::= {NEGative POSitive}
:SBUS<n>:SPI:CLOCK:TIMeout <time_value> (see page 1106)	:SBUS<n>:SPI:CLOCK:TIMeout? (see page 1106)	<time_value> ::= time in seconds in NR3 format
:SBUS<n>:SPI:DElay <value> (see page 1107)	:SBUS<n>:SPI:DElay? (see page 1107)	<value> ::= {OFF 2-63}
:SBUS<n>:SPI:FRAMing <value> (see page 1108)	:SBUS<n>:SPI:FRAMing? (see page 1108)	<value> ::= {CHIPselect {NCHIPselect NOTC} TIMeout}
:SBUS<n>:SPI:SOURce:CLOCK <source> (see page 1109)	:SBUS<n>:SPI:SOURce:CLOCK? (see page 1109)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:FRAME <source> (see page 1110)	:SBUS<n>:SPI:SOURce:FRAME? (see page 1110)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:SPI:SOURce:MISO <source> (see page 1111)	:SBUS<n>:SPI:SOURce:MISO? (see page 1111)	<value> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 124 :SBUS<n>:SPI Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:SPI:SOURce:MOSI <source> (see page 1112)	:SBUS<n>:SPI:SOURce:MOSI? (see page 1112)	<value> ::= {CHANnel<n> DIGital<d>} <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 page 1113)	:SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA? (see page 1113)	<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 page 1114)	:SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh? (see page 1114)	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA <string> (see page 1115)	:SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA? (see page 1115)	<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 page 1116)	:SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh? (see page 1116)	<width> ::= integer from 4 to 64 in NR1 format
:SBUS<n>:SPI:TRIGger:TYPE <value> (see page 1117)	:SBUS<n>:SPI:TRIGger:TYPE? (see page 1117)	<value> ::= {MOSI MISO}
:SBUS<n>:SPI:WIDTh <word_width> (see page 1118)	:SBUS<n>:SPI:WIDTh? (see page 1118)	<word_width> ::= integer 4-16 in NR1 format

:SBUS<n>:SPI:BITorder

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:BITorder <order>

<order> ::= {LSBFirst | MSBFirst}

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
• [":SBUS<n>:MODE"](#) on page 987
• [":SBUS<n>:SPI Commands"](#) on page 1102

:SBUS<n>:SPI:CLOCK:SLOPe

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:CLOCK:TIMEout"](#) on page 1106
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109

:SBUS<n>:SPI:CLOCK:TIMEout

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:CLOCK:TIMEout <time_value>

<time_value> ::= time in seconds in NR3 format

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.

Query Syntax :SBUS<n>:SPI:CLOCK:TIMEout?

The :SBUS<n>:SPI:CLOCK:TIMEout? query returns current SPI clock timeout setting.

Return Format <time_value><NL>

<time_value> ::= time in seconds in NR3 format

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:SPI:CLOCK:SLOPe"](#) on page 1105
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109
 - [":SBUS<n>:SPI:FRAMing"](#) on page 1108

:SBUS<n>:SPI:DELaY

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:DELaY <value>

<value> ::= {OFF | 2-63}

The :SBUS<n>:SPI:DELaY command specifies the number of bits to ignore (delay) before decoding the MISO stream.

Query Syntax :SBUS<n>:SPI:DELaY?

The :SBUS<n>:SPI:DELaY? query returns the specified number of delay bits or "OFF" if there are none.

Return Format <value><NL>

<value> ::= {OFF | 2-63}

See Also • [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111

:SBUS<n>:SPI:FRAMing

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:FRAMing <value>

<value> ::= {CHIPselect | {NCHipselect | NOTC} | TIMEout}

The :SBUS<n>:SPI:FRAMing command sets the SPI trigger framing value. If TIMEout is selected, the timeout value is set by the :SBUS<n>:SPI:CLOCK:TIMEout 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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:SPI:CLOCK:TIMEout"](#) on page 1106
 - [":SBUS<n>:SPI:SOURce:FRAME"](#) on page 1110

:SBUS<n>:SPI:SOURce:CLOCK

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:SOURce:CLOCK <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>:SPI:SOURce:CLOCK command sets the source for the SPI serial clock.

Query Syntax :SBUS<n>:SPI:SOURce:CLOCK?

The :SBUS<n>:SPI:SOURce:CLOCK? query returns the current source for the SPI serial clock.

Return Format <source><NL>

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:SPI:CLOCK:SLOPe"](#) on page 1105
 - [":SBUS<n>:SPI:CLOCK:TIMEout"](#) on page 1106
 - [":SBUS<n>:SPI:SOURce:FRAME"](#) on page 1110
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111

:SBUS<n>:SPI:SOURce:FRAME

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:SOURce:FRAME <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>: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 1479
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111
 - [":SBUS<n>:SPI:FRAMing"](#) on page 1108

:SBUS<n>:SPI:SOURce:MISO

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:SOURce:MISO <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>:SPI:SOURce:MISO command sets the source for the SPI serial MISO data.

Query Syntax :SBUS<n>:SPI:SOURce:MISO?

The :SBUS<n>:SPI:SOURce:MISO? query returns the current source for the SPI serial MISO data.

Return Format <source><NL>

- See Also**
- [":SBUS<n>:SPI:DELay"](#) on page 1107
 - ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109
 - [":SBUS<n>:SPI:SOURce:FRAME"](#) on page 1110
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA"](#) on page 1113
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA"](#) on page 1115
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh"](#) on page 1114
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh"](#) on page 1116

:SBUS<n>:SPI:SOURce:MOSI

N (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:SOURce:MOSI <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>:SPI:SOURce:MOSI command sets the source for the SPI serial MOSI data.

Query Syntax :SBUS<n>:SPI:SOURce:MOSI?

The :SBUS<n>:SPI:SOURce:MOSI? query returns the current source for the SPI serial MOSI data.

Return Format <source><NL>

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109
 - [":SBUS<n>:SPI:SOURce:FRAMe"](#) on page 1110
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA"](#) on page 1113
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA"](#) on page 1115
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh"](#) on page 1114
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh"](#) on page 1116

:SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTH"](#) on page 1114
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111

:SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA"](#) on page 1113
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111

:SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTH"](#) on page 1116
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112

:SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA"](#) on page 1115
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112

:SBUS<n>:SPI:TRIGger:TYPE

N (see [page 1708](#))

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 1479
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA"](#) on page 1113
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA"](#) on page 1115
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTh"](#) on page 1114
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTh"](#) on page 1116
 - [":TRIGger:MODE"](#) on page 1494

:SBUS<n>:SPI:WIDTh

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983

• [":SBUS<n>:MODE"](#) on page 987

• [":SBUS<n>:SPI Commands"](#) on page 1102

:SBUS<n>:UART Commands

NOTE

These commands are only valid when the UART/RS-232 triggering and serial decode option has been licensed.

Table 125 :SBUS<n>:UART Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:UART:BASE <base> (see page 1121)	:SBUS<n>:UART:BASE? (see page 1121)	<base> ::= {ASCIi BINary HEX}
:SBUS<n>:UART:BAUDrate <baudrate> (see page 1122)	:SBUS<n>:UART:BAUDrate? (see page 1122)	<baudrate> ::= integer from 100 to 8000000, 10000000, or 12000000
:SBUS<n>:UART:BITorder <bitorder> (see page 1123)	:SBUS<n>:UART:BITorder? (see page 1123)	<bitorder> ::= {LSBFirst MSBFirst}
:SBUS<n>:UART:FRAMing <value> (see page 1124)	:SBUS<n>:UART:FRAMing? (see page 1124)	<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 1125)	:SBUS<n>:UART:PARity? (see page 1125)	<parity> ::= {EVEN ODD NONE}
:SBUS<n>:UART:POLarity <polarity> (see page 1126)	:SBUS<n>:UART:POLarity? (see page 1126)	<polarity> ::= {HIGH LOW}
:SBUS<n>:UART:SOURce: RX <source> (see page 1127)	:SBUS<n>:UART:SOURce: RX? (see page 1127)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format

Table 125 :SBUS<n>:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:UART:SOURce:TX <source> (see page 1128)	:SBUS<n>:UART:SOURce:TX? (see page 1128)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:UART:TRIGger:BASE <base> (see page 1129)	:SBUS<n>:UART:TRIGger:BASE? (see page 1129)	<base> ::= {ASCii HEX}
:SBUS<n>:UART:TRIGger:BURSt <value> (see page 1130)	:SBUS<n>:UART:TRIGger:BURSt? (see page 1130)	<value> ::= {OFF 1 to 4096 in NR1 format}
:SBUS<n>:UART:TRIGger:DATA <value> (see page 1131)	:SBUS<n>:UART:TRIGger:DATA? (see page 1131)	<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 1132)	:SBUS<n>:UART:TRIGger:IDLE? (see page 1132)	<time_value> ::= time from 1 us to 10 s in NR3 format
:SBUS<n>:UART:TRIGger:QUALifier <value> (see page 1133)	:SBUS<n>:UART:TRIGger:QUALifier? (see page 1133)	<value> ::= {EQUal NOTequal GREaterthan LESSthan}
:SBUS<n>:UART:TRIGger:TYPE <value> (see page 1134)	:SBUS<n>:UART:TRIGger:TYPE? (see page 1134)	<value> ::= {RSTArt RSTOp RDATa RD1 RD0 RDX RXPARity TXPARity TSTArt TSTOp TDATa TD1 TD0 TDX}
:SBUS<n>:UART:WIDTh <width> (see page 1135)	:SBUS<n>:UART:WIDTh? (see page 1135)	<width> ::= {5 6 7 8 9}

:SBUS<n>:UART:BASE

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
• [":SBUS<n>:UART Commands"](#) on page 1119

:SBUS<n>:UART:BAUDrate

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:BAUDrate <baudrate>

<baudrate> ::= integer from 200 to 8000000, 10000000, or 12000000

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 in the range from 200 b/s to 8 Mb/s or to the specific values of 10 Mb/s or 12 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 200 to 8000000, 10000000, or 12000000

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:BITOrder

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:BITOrder <bitorder>

<bitorder> ::= {LSBFirst | MSBFirst}

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134
 - [":SBUS<n>:UART:SOURce:RX"](#) on page 1127
 - [":SBUS<n>:UART:SOURce:TX"](#) on page 1128

:SBUS<n>:UART:FRAMing

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:FRAMing <value>

<value> ::= {OFF | <decimal> | <nondecimal>}

<decimal> ::= 8-bit integer in decimal from 0-255 (0x00-0xff)

<nondecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

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.

Query Syntax :SBUS<n>:UART:FRAMing?

The :SBUS<n>:UART:FRAMing? query returns the current UART decode base setting.

Return Format <value><NL>

<value> ::= {OFF | <decimal>}

<decimal> ::= 8-bit integer in decimal from 0-255

Errors • ["-241, Hardware missing"](#) on page 1663

See Also • ["Introduction to :SBUS<n> Commands"](#) on page 983
 • [":SBUS<n>:UART Commands"](#) on page 1119
 • ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:SBUS<n>:UART:PARity

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:POLarity

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:SOURce:RX

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:SOURce:RX <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>: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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134
 - [":SBUS<n>:UART:BITOrder"](#) on page 1123

:SBUS<n>:UART:SOURce:TX

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:SOURce:TX <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>: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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134
 - [":SBUS<n>:UART:BITOrder"](#) on page 1123

:SBUS<n>:UART:TRIGger:BASE

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:DATA"](#) on page 1131

:SBUS<n>:UART:TRIGger:BURSt

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:IDLE"](#) on page 1132
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:TRIGger:IDLE

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:BURSt"](#) on page 1130
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:TRIGger:QUALifier

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:UART:TRIGger:TYPE

N (see [page 1708](#))

Command Syntax :SBUS<n>:UART:TRIGger:TYPE <value>

<value> ::= {RSTArt | RSTOp | RDATA | RD1 | RD0 | RDX | RXParity
| TXParity | TSTArt | TSTOp | TDATA | 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 | RXPA | TXPA | TSTA
| TSTO | TDAT | TD1 | TD0 | TDX}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:DATA"](#) on page 1131
 - [":SBUS<n>:UART:TRIGger:QUALifier"](#) on page 1133
 - [":SBUS<n>:UART:WIDTH"](#) on page 1135

:SBUS<n>:UART:WIDTh

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":SBUS<n>:UART:TRIGger:TYPE"](#) on page 1134

:SBUS<n>:USB Commands

NOTE

These commands are only valid when a USB 2.0 triggering and serial decode option has been licensed.

Table 126 :SBUS<n>:USB Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:USB:BASE <base> (see page 1139)	:SBUS<n>:USB:BASE? (see page 1139)	<base> ::= {ASCIi BINary DECimal HEX}
:SBUS<n>:USB:SOURce:D MINus <source> (see page 1140)	:SBUS<n>:USB:SOURce:D MINus? (see page 1140)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:USB:SOURce:D PLus <source> (see page 1141)	:SBUS<n>:USB:SOURce:D PLus? (see page 1141)	<source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:SBUS<n>:USB:SOURce:D IFFerential <source> (see page 1142)	:SBUS<n>:USB:SOURce:D IFFerential? (see page 1142)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:USB:SPEEd <speed> (see page 1143)	:SBUS<n>:USB:SPEEd? (see page 1143)	<speed> ::= {LOW FULL HIGH}
:SBUS<n>:USB:TRIGger <condition> (see page 1144)	:SBUS<n>:USB:TRIGger? (see page 1144)	<condition> ::= {SOP EOP ENTersuspend EXITSuspend RESet TOKen DATA HANDshake SPECial ALLerrors PIDerror CRC5error CRC16error GLITCherror STUFFerror SE1error}
:SBUS<n>:USB:TRIGger: ADDReSS <string> (see page 1145)	:SBUS<n>:USB:TRIGger: ADDReSS? (see page 1145)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}

Table 126 :SBUS<n>:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:USB:TRIGger: CRC <string> (see page 1146)	:SBUS<n>:USB:TRIGger: CRC? (see page 1146)	<string> ::= "nnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: DATA <string> (see page 1147)	:SBUS<n>:USB:TRIGger: DATA? (see page 1147)	<string> ::= "nnnn..." where n ::= {0 1 X} <string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: DATA:LENGth <value> (see page 1148)	:SBUS<n>:USB:TRIGger: DATA:LENGth? (see page 1148)	<length> ::= data length between 1-20
:SBUS<n>:USB:TRIGger: ENDPoint <string> (see page 1149)	:SBUS<n>:USB:TRIGger: ENDPoint? (see page 1149)	<string> ::= "nnnn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: ET <string> (see page 1150)	:SBUS<n>:USB:TRIGger: ET? (see page 1150)	<string> ::= "nn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0 1 2 3 X \$}
:SBUS<n>:USB:TRIGger: FRaMe <string> (see page 1151)	:SBUS<n>:USB:TRIGger: FRaMe? (see page 1151)	<string> ::= "nnnnnnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnnnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: HADDress <string> (see page 1152)	:SBUS<n>:USB:TRIGger: HADDress? (see page 1152)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: PCHeck <string> (see page 1153)	:SBUS<n>:USB:TRIGger: PCHeck? (see page 1153)	<string> ::= "nnnn" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger: PID:DATA <pid> (see page 1154)	:SBUS<n>:USB:TRIGger: PID:DATA? (see page 1154)	<pid> ::= {DATA0 DATA1 DATA2 MDATa}
:SBUS<n>:USB:TRIGger: PID:HANDshake <pid> (see page 1155)	:SBUS<n>:USB:TRIGger: PID:HANDshake? (see page 1155)	<pid> ::= {ACK NAK STAl1 NYET}
:SBUS<n>:USB:TRIGger: PID:SPECIAL <pid> (see page 1156)	:SBUS<n>:USB:TRIGger: PID:SPECIAL? (see page 1156)	<pid> ::= {PING PRE ERR SPLit}

Table 126 :SBUS<n>:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SBUS<n>:USB:TRIGger:PID:TOKE n <pid> (see page 1157)	:SBUS<n>:USB:TRIGger:PID:TOKE n? (see page 1157)	<pid> ::= {OUT IN SETup SOF}
:SBUS<n>:USB:TRIGger:PORT <string> (see page 1158)	:SBUS<n>:USB:TRIGger:PORT? (see page 1158)	<string> ::= "nnnnnnnn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SBUS<n>:USB:TRIGger:SC <string> (see page 1159)	:SBUS<n>:USB:TRIGger:SC? (see page 1159)	<string> ::= "n" where n ::= {0 1 X} <string> ::= "0xn" where n ::= {0 1 X \$}
:SBUS<n>:USB:TRIGger:SEU <string> (see page 1160)	:SBUS<n>:USB:TRIGger:SEU? (see page 1160)	<string> ::= "nn" where n ::= {0 1 X} <string> ::= "0xnn" where n ::= {0 1 2 3 X \$}

:SBUS<n>:USB:BASE

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:BASE <base>

<base> ::= {ASCIi | BINary | DECimal | HEX}

The :SBUS<n>:USB:BASE command determines the base to use for the USB decode and Lister display, controlling how the data or payload field is displayed.

All other fields are displayed in hex

Query Syntax :SBUS<n>:USB:BASE?

The :SBUS<n>:USB:BASE? query returns the currently specified base.

Return Format <base><NL>

<base> ::= {ASC | BIN | DEC | HEX}

:SBUS<n>:USB:SOURce:DMINus

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:SOURce:DMINus <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>:USB:SOURce:DMINus command specifies which signal is used as the USB D- source for the selected bus.

The D- source is only used for Low and Full Speed USB.

Query Syntax :SBUS<n>:USB:SOURce:DMINus?

The :SBUS<n>:USB:SOURce:DMINus? query returns the specified D- source.

Return Format <source><NL>

<source> ::= {CHAN<n> | DIG<d>}

- See Also**
- [":SBUS<n>:USB:SOURce:DPLus"](#) on page 1141
 - [":SBUS<n>:USB:SPEed"](#) on page 1143

:SBUS<n>:USB:SOURce:DPLus

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:SOURce:DPLus <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>:USB:SOURce:DPLus command specifies which signal is used as the USB D+ source for the selected bus.

The D+ source is only used for Low and Full Speed USB.

Query Syntax :SBUS<n>:USB:SOURce:DPLus?

The :SBUS<n>:USB:SOURce:DPLus? query returns the specified D+ source.

Return Format <source><NL>

<source> ::= {CHAN<n> | DIG<d>}

- See Also**
- [":SBUS<n>:USB:SOURce:DMINus"](#) on page 1140
 - [":SBUS<n>:USB:SPEed"](#) on page 1143

:SBUS<n>:USB:SOURce:DIFFerential

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:SOURce:DIFFerential <source>

<source> ::= {CHANnel<n>}

<n> ::= 1 to (# analog channels) in NR1 format

The :SBUS<n>:USB:SOURce:DIFFerential command specifies which signal is used as the differential source for the selected bus.

Differential sources are only used for High Speed USB.

Query Syntax :SBUS<n>:USB:SOURce:DIFFerential?

The :SBUS<n>:USB:SOURce:DIFFerential? query returns the specified differential source.

Return Format <source><NL>

<source> ::= {CHAN<n>}

See Also • [":SBUS<n>:USB:SPEed"](#) on page 1143

:SBUS<n>:USB:SPEEd

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:SPEEd <speed>

<speed> ::= {LOW | FULL | HIGH}

The :SBUS<n>:USB:SPEEd command specifies the speed of the USB interface for the selected bus.

Query Syntax :SBUS<n>:USB:SPEEd?

The :SBUS<n>:USB:SPEEd? query returns the speed setting.

Return Format <speed><NL>

<speed> ::= {LOW | FULL | HIGH}

- See Also**
- [":SBUS<n>:USB:SOURce:DPLus"](#) on page 1141
 - [":SBUS<n>:USB:SOURce:DMINus"](#) on page 1140
 - [":SBUS<n>:USB:SOURce:DIFFerential"](#) on page 1142

:SBUS<n>:USB:TRIGger

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger <condition>

```
<condition> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet
                  | TOKen | DATA | HANDshake | SPECial | ALLerrors
                  | PIDerror | CRC5error | CRC16error | GLITCherror
                  | STUFFerror | SE1error}
```

The :SBUS<n>:USB:TRIGger command specifies the USB trigger mode for the selected bus.

Query Syntax :SBUS<n>:USB:TRIGger?

The :SBUS<n>:USB:TRIGger? query returns the specified USB trigger mode.

Return Format <condition><NL>

```
<condition> ::= {SOP | EOP | ENT | EXIT | RES | TOK | DATA | HAND
                  | SPEC | ALL | PID | CRC5 | CRC16 | GLIT | STUFF
                  | SE1}
```

- See Also**
- [":SBUS<n>:USB:TRIGger:PID:DATA"](#) on page 1154
 - [":SBUS<n>:USB:TRIGger:PID:HANDshake"](#) on page 1155
 - [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156
 - [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:ADDRess

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:ADDRess <string>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:ADDRess command specifies the 7-bit Address portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:ADDRess?

The :SBUS<n>:USB:TRIGger:ADDRess? query returns the specified Address portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

See Also

- [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156
- [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:CRC

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:CRC <string>

<string> ::= "nnnnn" where n ::= {0 | 1 | X}

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:CRC command specifies the CRC portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:CRC?

The :SBUS<n>:USB:TRIGger:CRC? query returns the specified CRC portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnnn" where n ::= {0 | 1 | X}

- See Also**
- [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156
 - [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:DATA <string>

<string> ::= "nnnn..." where n ::= {0 | 1 | X}

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:DATA command specifies the Data portion of the trigger for the selected bus, in binary or hex.

See the :SBUS<n>:USB:TRIGger:DATA:LENGth command for setting the length of the data value.

Query Syntax :SBUS<n>:USB:TRIGger:DATA?

The :SBUS<n>:USB:TRIGger:DATA? query returns the specified Data portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnn..." where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:DATA:LENGth"](#) on page 1148

:SBUS<n>:USB:TRIGger:DATA:LENGth

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:DATA:LENGth <length>

<length> ::= data length between 1-20

The :SBUS<n>:USB:TRIGger:DATA:LENGth command specifies the data length in bytes.

Query Syntax :SBUS<n>:USB:TRIGger:DATA:LENGth?

The :SBUS<n>:USB:TRIGger:DATA:LENGth? query returns the specified data length.

Return Format <length><NL>

<length> ::= data length between 1-20

See Also • [":SBUS<n>:USB:TRIGger:DATA"](#) on page 1147

:SBUS<n>:USB:TRIGger:ENDPoint

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:ENDPoint <string>

<string> ::= "nnnn" where n ::= {0 | 1 | x}

<string> ::= "0xn" where n ::= {0,...,9 | A,...,F | x | \$}

The :SBUS<n>:USB:TRIGger:ENDPoint command specifies the 4-bit Endpoint portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:ENDPoint?

The :SBUS<n>:USB:TRIGger:ENDPoint? query returns the specified Endpoint portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnn" where n ::= {0 | 1 | x}

- See Also**
- [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156
 - [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:ET

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:ET <string>

<string> ::= "nn" where n ::= {0 | 1 | X}

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

The :SBUS<n>:USB:TRIGger:ET command specifies the 2-bit ET portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:ET?

The :SBUS<n>:USB:TRIGger:ET? query returns the specified ET portion of the trigger.

Return Format <string><NL>

<string> ::= "nn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156

:SBUS<n>:USB:TRIGger:FRAME

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:FRAME <string>

<string> ::= "nnnnnnnnnnnn" where n ::= {0 | 1 | X}

<string> ::= "0xnnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:FRAME command specifies the 11-bit Frame portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:FRAME?

The :SBUS<n>:USB:TRIGger:FRAME? query returns the specified Frame portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnnnnnnnnnn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:HADDress

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:HADDress <string>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:HADDress command specifies the 7-bit Hub Address portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:HADDress?

The :SBUS<n>:USB:TRIGger:HADDress? query returns the specified Hub Address portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156

:SBUS<n>:USB:TRIGger:PCHeck

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PCHeck <string>

<string> ::= "nnnn" where n ::= {0 | 1 | x}

<string> ::= "0xn" where n ::= {0,...,9 | A,...,F | x | \$}

The :SBUS<n>:USB:TRIGger:PCHeck command specifies the 4-bit PID check portion of the trigger for the selected bus, in binary or hex.

See the :SBUS<n>:USB:TRIGger:PID commands for setting the PID.

Query Syntax :SBUS<n>:USB:TRIGger:PCHeck?

The :SBUS<n>:USB:TRIGger:PCHeck? query returns the specified PID check portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnn" where n ::= {0 | 1 | x}

- See Also**
- [":SBUS<n>:USB:TRIGger:PID:DATA"](#) on page 1154
 - [":SBUS<n>:USB:TRIGger:PID:HANDshake"](#) on page 1155
 - [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156
 - [":SBUS<n>:USB:TRIGger:PID:TOKen"](#) on page 1157

:SBUS<n>:USB:TRIGger:PID:DATA

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PID:DATA <pid>

<pid> ::= {DATA0 | DATA1 | DATA2 | MDATA}

The :SBUS<n>:USB:TRIGger:PID:DATA command specifies the USB data PID to trigger on for the selected bus.

The specified PID does not include the PID check value, which is specified using the :SBUS<n>:USB:TRIGger:PIDCheck command.

The DATA USB trigger mode can be selected using the :SBUS<n>:USB:TRIGger command.

Query Syntax :SBUS<n>:USB:TRIGger:PID:DATA?

The :SBUS<n>:USB:TRIGger:PID:DATA? query returns the specified data PID.

Return Format <pid><NL>

<pid> ::= {DATA0 | DATA1 | DATA2 | MDATA}

- See Also**
- [":SBUS<n>:USB:TRIGger:PIDCheck"](#) on page 1153
 - [":SBUS<n>:USB:TRIGger"](#) on page 1144

:SBUS<n>:USB:TRIGger:PID:HANDshake

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PID:HANDshake <pid>

<pid> ::= {ACK | NAK | STAL1 | NYET}

The :SBUS<n>:USB:TRIGger:PID:HANDshake command specifies the USB handshake PID to trigger on for the selected bus.

The specified PID does not include the PID check value, which is specified using the :SBUS<n>:USB:TRIGger:PIDCheck command.

The HANDshake USB trigger mode can be selected using the :SBUS<n>:USB:TRIGger command.

Query Syntax :SBUS<n>:USB:TRIGger:PID:HANDshake?

The :SBUS<n>:USB:TRIGger:PID:HANDshake? query returns the specified handshake PID.

Return Format <pid><NL>

<pid> ::= {ACK | NAK | STAL | NYET}

- See Also**
- [":SBUS<n>:USB:TRIGger:PCHeck"](#) on page 1153
 - [":SBUS<n>:USB:TRIGger"](#) on page 1144

:SBUS<n>:USB:TRIGger:PID:SPEcial

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PID:SPEcial <pid>

<pid> ::= {PING | PRE | ERR | SPLit}

The :SBUS<n>:USB:TRIGger:PID:SPEcial command specifies the USB special PID to trigger on for the selected bus.

The specified PID does not include the PID check value, which is specified using the :SBUS<n>:USB:TRIGger:PIDCheck command.

The SPEcial USB trigger mode can be selected using the :SBUS<n>:USB:TRIGger command.

Query Syntax :SBUS<n>:USB:TRIGger:PID:SPEcial?

The :SBUS<n>:USB:TRIGger:PID:SPEcial? query returns the specified special PID.

Return Format <pid><NL>

<pid> ::= {PING | PRE | ERR | SPL}

- See Also**
- [":SBUS<n>:USB:TRIGger:PCheck"](#) on page 1153
 - [":SBUS<n>:USB:TRIGger"](#) on page 1144

:SBUS<n>:USB:TRIGger:PID:TOKen

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PID:TOKen <pid>

<pid> ::= {OUT | IN | SETup | SOF}

The :SBUS<n>:USB:TRIGger:PID:TOKen command specifies the USB token PID to trigger on for the selected bus.

The specified PID does not include the PID check value, which is specified using the :SBUS<n>:USB:TRIGger:PIDCheck command.

The TOKen USB trigger mode can be selected using the :SBUS<n>:USB:TRIGger command.

Query Syntax :SBUS<n>:USB:TRIGger:PID:TOKen?

The :SBUS<n>:USB:TRIGger:PID:TOKen? query returns the specified token PID.

Return Format <pid><NL>

<pid> ::= {OUT | IN | SETup | SOF}

- See Also**
- [":SBUS<n>:USB:TRIGger:PIDCheck"](#) on page 1153
 - [":SBUS<n>:USB:TRIGger"](#) on page 1144

:SBUS<n>:USB:TRIGger:PORT

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:PORT <string>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SBUS<n>:USB:TRIGger:PORT command specifies the 7-bit Port portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:PORT?

The :SBUS<n>:USB:TRIGger:PORT? query returns the specified Port portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnnnnnn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156

:SBUS<n>:USB:TRIGger:SC

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:SC <string>

<string> ::= "n" where n ::= {0 | 1 | X}

<string> ::= "0xn" where n ::= {0 | 1 | X | \$}

The :SBUS<n>:USB:TRIGger:SC command specifies the 1-bit SC portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:SC?

The :SBUS<n>:USB:TRIGger:SC? query returns the specified SC portion of the trigger.

Return Format <string><NL>

<string> ::= "nnnn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156

:SBUS<n>:USB:TRIGger:SEU

N (see [page 1708](#))

Command Syntax :SBUS<n>:USB:TRIGger:SEU <string>

<string> ::= "nn" where n ::= {0 | 1 | X}

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

The :SBUS<n>:USB:TRIGger:SEU command specifies the 2-bit S and E or U portion of the trigger for the selected bus, in binary or hex.

Query Syntax :SBUS<n>:USB:TRIGger:SEU?

The :SBUS<n>:USB:TRIGger:SEU? query returns the specified S and E or U portion of the trigger.

Return Format <string><NL>

<string> ::= "nn" where n ::= {0 | 1 | X}

See Also • [":SBUS<n>:USB:TRIGger:PID:SPECial"](#) on page 1156

:SBUS<n>:USBPd Commands

NOTE

These commands are valid when the USB PD (Power Delivery) serial decode and triggering option has been licensed.

Table 127 :SBUS<n>:USBPd Commands Summary

Command	Query	Options and Query Returns
:SBUS<n>:USBPd:SOURce <source> (see page 1162)	:SBUS<n>:USBPd:SOURce ? (see page 1162)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format
:SBUS<n>:USBPd:TRIGger <mode> (see page 1163)	:SBUS<n>:USBPd:TRIGger? (see page 1163)	<mode> ::= {PSTart EOP SOP SPRime SDPRime SPDebug SDPDebug HRST CRST CRCerror PERRor HEADer}
:SBUS<n>:USBPd:TRIGger:HEADer <type> (see page 1164)	:SBUS<n>:USBPd:TRIGger:HEADer? (see page 1164)	<type> ::= {CMESsage DMESsage EMESsage VALue}
:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage <type> (see page 1166)	:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage? (see page 1166)	<type> ::= {GOODcrc GOTOmin ACCEpt REJect PING PSRDy GSRCap GSNCap DRSwap PRSWap VCSwap WAIT SRST GSCX GSTatus FRSwap GPSTatus GCCodes}
:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage <type> (see page 1168)	:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage? (see page 1168)	<type> ::= {SRCap REQuest BIST SNCap BSTatus ALERt GCINfo VDEfined}
:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage <type> (see page 1169)	:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage? (see page 1169)	<type> ::= {SCX STATus GBCap GBSTatus BCAP GMINfo MINfo SREQuest SRESponse FREQuest FRESponse PSTatus CINfo CCODEs}
:SBUS<n>:USBPd:TRIGger:HEADer:VALue <string> (see page 1171)	:SBUS<n>:USBPd:TRIGger:HEADer:VALue? (see page 1171)	<string> ::= "nn...n" where n ::= {0 1 X} <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X}
:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier <type> (see page 1172)	:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier? (see page 1172)	<type> ::= {NONE SOP SPRime SDPRime}

:SBUS<n>:USBPd:SOURce

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:SOURce <source>

<source> ::= {CHANnel<n>}

<n> ::= 1 to (# analog channels) in NR1 format

The :SBUS<n>:USBPd:SOURce command selects the USB PD waveform source. You can use analog channels.

Query Syntax :SBUS<n>:USBPd:SOURce?

The :SBUS<n>:USBPd:SOURce? query returns the selected analog input channel.

Return Format <source><NL>

<source> ::= {CHAN<n>}

- See Also**
- [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171
 - [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger <mode>

```
<mode> ::= {PSTart | EOP | SOP | SPRime | SDPRime | SPDebug | SDPDebug
            | HRST | CRST | CRCerror | PERRor | HEADer}
```

The :SBUS<n>:USBPd:TRIGger command selects the USB PD trigger mode:

- PSTart – **Preamble Start**, triggers at the start of a preamble, which starts with 0.
- EOP – **EOP**, triggers at the end of packet.
- SOP – **SOP**, triggers on ordered set Sync-1, Sync-1, Sync-1, Sync-2.
- SPRime – **SOP'**, triggers on ordered set Sync-1, Sync-1, Sync-3, Sync-3.
- SDPRime – **SOP"**, triggers on ordered set Sync-1, Sync-3, Sync-1, Sync-3.
- SPDebug – **SOP' Debug**, triggers on ordered set Sync-1, RST-2, RST-2, Sync-3.
- SDPDebug – **SOP" Debug**, triggers on ordered set Sync-1, RST-2, Sync-3, Sync-2.
- HRST – **Hard Reset**, triggers on ordered set RST-1, RST-1, RST-1, RST-2.
- CRST – **Cable Reset**, triggers on ordered set RST-1, Sync-1, RST-1, Sync-3.
- CRCerror – **CRC Error**, triggers when an error is detected on a 32-bit CRC.
- PERRor – **Preamble Error**, triggers when an error is detected on a 64-bit sequence of alternating 0 and 1.
- HEADer – **Header Content**, triggers on a user-defined 16-bit value.

In this mode, use the :SBUS<n>:USBPd:TRIGger:HEADer command to select the header type.

Query Syntax :SBUS<n>:USBPd:TRIGger?

The :SBUS<n>:USBPd:TRIGger? query returns the selected USB PD trigger mode.

Return Format <mode><NL>

```
<mode> ::= {PST | EOP | SOP | SPR | SDPR | SPD | SDPD | HRST | CRST
            | CRC | PERR | HEAD}
```

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171
 - [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer <type>

<type> ::= {CMESsage | DMESsage | EMESsage | VALue}

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command), the :SBUS<n>:USBPd:TRIGger:HEADer command selects the header type:

- CMESsage – **Control Message**, triggers on control message types (0 data object).

When the CMESsage header type is selected, use the :SBUS<n>:USBPd:TRIGger:HEADer:CMESsage command to select the control message type.

- DMESsage – **Data Message**, triggers on data message types (1 or more data objects).

When the DMESsage header type is selected, use the :SBUS<n>:USBPd:TRIGger:HEADer:DMESsage command to select the data message type.

- EMESsage – **Extended Message**, triggers on extended message types (bit 15 is set).

When the EMESsage header type is selected, use the :SBUS<n>:USBPd:TRIGger:HEADer:EMESsage command to select the extended message type.

- VALue – **Value**, triggers on a user-defined header value.

When the CMESsage header type is selected, use the :SBUS<n>:USBPd:TRIGger:HEADer:VALue command to specify the user-defined header value.

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer?

The :SBUS<n>:USBPd:TRIGger:HEADer? query returns the selected header type.

Return Format <type><NL>

<type> ::= {CMES | DMES | EMES | VAL}

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171

- **":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"** on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer:CMESsage

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer:CMESsage <type>

```
<type> ::= {GOODcrc | GOTOmin | ACcept | REJect | PING | PSRDy
            | GSRCap | GSNCap | DRSwap | PRSwap | VCSwap | WAIT | SRST | GSCX
            | GSTatus | FRSwap | GPSTatus | GCCodes}
```

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command) and the **Control Message** header type is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage" command), the :SBUS<n>:USBPd:TRIGger:HEADer:CMESsage command selects the control message type:

- GOODcrc – GoodCRC
- GOTOmin – GotoMin
- ACcept – Accept
- REJect – Reject
- PING – Ping
- PSRDy – PS_RDY
- GSRCap – Get_Source_Cap
- GSNCap – Get_Sink_Cap
- DRSwap – DR_Swap
- PRSwap – PR_Swap
- VCSwap – VCONN_Swap
- WAIT – Wait
- SRST – Soft_Reset
- GSCX – Get_Source_Cap_Extended
- GSTatus – Get_Status
- FRSwap – FR_Swap
- GPSTatus – Get_PPS_Status
- GCCodes – Get_Country_Codes

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer:CMESsage?

The :SBUS<n>:USBPd:TRIGger:HEADer:CMESsage? query returns the selected control message type.

Return Format <type><NL>

```
<type> ::= {GOOD | GOTO | ACC | REJ | PING | PSRD | GSRC | GSNC | DRSW
            | PRSW | VCSW | WAIT | SRST | GSCX | GST | FRSW | GPST | GCC}
```

- See Also
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171
 - [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer:DMESsage

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer:DMESsage <type>

<type> ::= {SRCap | REQuest | BIST | SNCap | BStatus | ALERt | GCINfo
| VDEFined}

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command) and the **Data Message** header type is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage" command), the :SBUS<n>:USBPd:TRIGger:HEADer:DMESsage command selects the data message type:

- SRCap – Source_Capabilities
- REQuest – Request
- BIST – BIST
- SNCap – Sink_Capabilities
- BStatus – Battery_Status
- ALERt – Alert
- GCINfo – Get_Country_Info
- VDEFined – Vendor_Defined

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer:DMESsage?

The :SBUS<n>:USBPd:TRIGger:HEADer:DMESsage? query returns the selected data message type.

Return Format <type><NL>

<type> ::= {SRC | REQ | BIST | SNC | BST | ALER | GCIN | VDEF}

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171
 - [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer:EMESsage

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer:EMESsage <type>

```
<type> ::= {SCX | STATus | GBCap | GBSTatus | BCAP | GMINfo | MINFo
           | SREQuest | SRESponse | FREQuest | FRESponse | PSTatus | CINFo
           | CCODes}
```

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command) and the **Extended Message** header type is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage" command), the :SBUS<n>:USBPd:TRIGger:HEADer:EMESsage command selects the extended message type:

- SCX – Source_Capabilities_Extended
- STATus – Status
- GBCap – Get_Battery_Cap
- GBSTatus – Get_Battery_Status
- BCAP – Battery_Capabilities
- GMINfo – Get_Manufacturer_Info
- MINFo – Manufacturer_Info
- SREQuest – Security_Request
- SRESponse – Security_Response
- FREQuest – Firmware_Update_Request
- FRESponse – Firmware_Update_Response
- PSTatus – PPS_Status
- CINFo – Country_Info
- CCODes – Country_Codes

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer:EMESsage?

The :SBUS<n>:USBPd:TRIGger:HEADer:EMESsage? query returns the selected extended message type

Return Format <type><NL>

```
<type> ::= {SCX | STATus | GBC | GBST | BCAP | GMIN | MINF | SREQ
           | SRES | FREQ | FRES | PSTatus | CINF | CCOD}
```

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166

- [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
- [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171
- [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer:VALue

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer:VALue <string>

<string> ::= "nn...n" where n ::= {0 | 1 | X}

<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X}

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command) and the user-defined **Value** type is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer:VALue" command), the :SBUS<n>:USBPd:TRIGger:HEADer:VALue command specifies the user-defined header value.

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer:VALue?

The :SBUS<n>:USBPd:TRIGger:HEADer:VALue? query returns the specified user-defined header value.

Returned data values are always quoted binary format strings.

Return Format <string><NL>

<string> ::= "nn...n" where n ::= {0 | 1 | X}

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:QUALifier"](#) on page 1172

:SBUS<n>:USBPd:TRIGger:HEADer:QUALifier

N (see [page 1708](#))

Command Syntax :SBUS<n>:USBPd:TRIGger:HEADer:QUALifier <type>

<type> ::= {NONE | SOP | SPRime | SDPRime}

When the **Header Content** trigger mode is selected (with the ":SBUS<n>:USBPd:TRIGger:HEADer" command), the :SBUS<n>:USBPd:TRIGger:HEADer:QUALifier command selects an additional qualifier for the Header Content trigger:

- NONE – **None**. There is no additional qualifier for the trigger.
- SOP – **SOP**. The trigger occurs on Sync-1, Sync-1, Sync-1, Sync-2 ordered sets only.
- SPRime – **SOP'**. The trigger occurs on Sync-1, Sync-1, Sync-3, Sync-3 ordered sets only.
- SDPRime – **SOP"**. The trigger occurs on Sync-1, Sync-3, Sync-1, Sync-3 ordered sets only.

Query Syntax :SBUS<n>:USBPd:TRIGger:HEADer:QUALifier?

The :SBUS<n>:USBPd:TRIGger:HEADer:QUALifier? query returns the selected header content trigger qualifier.

Return Format <type><NL>

<type> ::= {NONE | SOP | SPRime | SDPRime}

- See Also**
- [":SBUS<n>:USBPd:SOURce"](#) on page 1162
 - [":SBUS<n>:USBPd:TRIGger"](#) on page 1163
 - [":SBUS<n>:USBPd:TRIGger:HEADer"](#) on page 1164
 - [":SBUS<n>:USBPd:TRIGger:HEADer:CMESsage"](#) on page 1166
 - [":SBUS<n>:USBPd:TRIGger:HEADer:DMESsage"](#) on page 1168
 - [":SBUS<n>:USBPd:TRIGger:HEADer:EMESsage"](#) on page 1169
 - [":SBUS<n>:USBPd:TRIGger:HEADer:VALue"](#) on page 1171

36 :SEARch Commands

Control the event search modes and parameters for each search type. See:

- **"General :SEARch Commands"** on page 1174
- **":SEARch:EDGE Commands"** on page 1179
- **":SEARch:GLITCh Commands"** on page 1182 (Pulse Width search)
- **":SEARch:PEAK Commands"** on page 1189
- **":SEARch:RUNT Commands"** on page 1194
- **":SEARch:TRANSition Commands"** on page 1199
- **":SEARch:SERial:A429 Commands"** on page 1204
- **":SEARch:SERial:CAN Commands"** on page 1210
- **":SEARch:SERial:IIC Commands"** on page 1220
- **":SEARch:SERial:LIN Commands"** on page 1227
- **":SEARch:SERial:M1553 Commands"** on page 1236
- **":SEARch:SERial:SENT Commands"** on page 1240
- **":SEARch:SERial:SPI Commands"** on page 1245
- **":SEARch:SERial:UART Commands"** on page 1249
- **":SEARch:SERial:USB Commands"** on page 1254
- **":SEARch:SERial:USBPd Commands"** on page 1272

General :SEARCH Commands

Table 128 General :SEARCH Commands Summary

Command	Query	Options and Query Returns
n/a	:SEARCH:COUNT? (see page 1175)	<count> ::= an integer count value
:SEARCH:EVENT <event_number> (see page 1176)	:SEARCH:EVENT? (see page 1176)	<event_number> ::= the integer number of a found search event
:SEARCH:MODE <value> (see page 1177)	:SEARCH:MODE? (see page 1177)	<value> ::= {EDGE GLITCh RUNT TRANSition SERIAL{1 2} PEAK}
:SEARCH:STATE <value> (see page 1178)	:SEARCH:STATE? (see page 1178)	<value> ::= {{0 OFF} {1 ON}}

:SEARCh:COUNT?

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:EVENT"](#) on page 1176
 - [":SEARCh:STATe"](#) on page 1178
 - [":SEARCh:MODE"](#) on page 1177

:SEARCh:EVENT

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:COUNt?"](#) on page 1175
 - [":SEARCh:STAtE"](#) on page 1178
 - [":SEARCh:MODE"](#) on page 1177

:SEARCh:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:MODE <value>

<value> ::= {EDGE | GLITCh | RUNT | TRANSition | SERIAL{1 | 2} | PEAK}

The :SEARCh:MODE command selects the search mode.

The command is only valid when the :SEARCh:STAtE is ON.

Query Syntax :SEARCh:MODE?

The :SEARCh:MODE? query returns the currently selected mode or OFF if the :SEARCh:STAtE is OFF.

Return Format <value><NL>

<value> ::= {EDGE | GLIT | RUNT | TRAN | SER{1 | 2} | PEAK | OFF}

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:STAtE"](#) on page 1178
 - [":SEARCh:COUNt?"](#) on page 1175
 - [":SEARCh:EVENTt"](#) on page 1176

:SEARCh:STATe

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:COUNt?"](#) on page 1175
 - [":SEARCh:EVENT"](#) on page 1176

:SEARCh:EDGE Commands

Table 129 :SEARCh:EDGE Commands Summary

Command	Query	Options and Query Returns
:SEARCh:EDGE:SLOPe <slope> (see page 1180)	:SEARCh:EDGE:SLOPe? (see page 1180)	<slope> ::= {POSitive NEGative EITHer}
:SEARCh:EDGE:SOURce <source> (see page 1181)	:SEARCh:EDGE:SOURce? (see page 1181)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

:SEARCh:EDGE:SLOPe

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173

:SEARCh:EDGE:SOURce

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173

:SEARCH:GLITCh Commands

Table 130 :SEARCH:GLITCh Commands Summary

Command	Query	Options and Query Returns
:SEARCH:GLITCh:GREate rthan <greater_than_time>[s uffix] (see page 1183)	:SEARCH:GLITCh:GREate rthan? (see page 1183)	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:SEARCH:GLITCh:LESSt han <less_than_time>[suff ix] (see page 1184)	:SEARCH:GLITCh:LESSt han? (see page 1184)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:SEARCH:GLITCh:POLari ty <polarity> (see page 1185)	:SEARCH:GLITCh:POLari ty? (see page 1185)	<polarity> ::= {POSitive NEGative}
:SEARCH:GLITCh:QUALif ier <qualifier> (see page 1186)	:SEARCH:GLITCh:QUALif ier? (see page 1186)	<qualifier> ::= {GREaterthan LESSthan RANGE}
:SEARCH:GLITCh:RANGE <less_than_time>[suff ix], <greater_than_time>[s uffix] (see page 1187)	:SEARCH:GLITCh:RANGE? (see page 1187)	<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 page 1188)	:SEARCH:GLITCh:SOURce ? (see page 1188)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format

:SEARCh:GLITCh:GREaterthan

N (see [page 1708](#))

- 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 36](#), “:SEARCh Commands,” starting on page 1173
 • [":SEARCh:GLITCh:SOURce"](#) on page 1188
 • [":SEARCh:GLITCh:QUALifier"](#) on page 1186
 • [":SEARCh:MODE"](#) on page 1177

:SEARCh:GLITCh:LESSthan

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
- [":SEARCh:GLITCh:SOURce"](#) on page 1188
- [":SEARCh:GLITCh:QUALifier"](#) on page 1186
- [":SEARCh:MODE"](#) on page 1177

:SEARCh:GLITCh:POLarity

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:GLITCh:SOURce"](#) on page 1188

:SEARCh:GLITCh:QUALifier

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:GLITCh:SOURce"](#) on page 1188
 - [":SEARCh:MODE"](#) on page 1177

:SEARCh:GLITCh:SOURce

N (see [page 1708](#))

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 36](#), ":SEARCh Commands," starting on page 1173
 - [Chapter 36](#), ":SEARCh Commands," starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:GLITCh:POLarity"](#) on page 1185
 - [":SEARCh:GLITCh:QUALifier"](#) on page 1186
 - [":SEARCh:GLITCh:RANGe"](#) on page 1187

:SEARCh:PEAK Commands

Table 131 :SEARCh:PEAK Commands Summary

Command	Query	Options and Query Returns
:SEARCh:PEAK:EXCURsion <delta_level> (see page 1190)	:SEARCh:PEAK:EXCURsion? (see page 1190)	<delta_level> ::= required change in level to be recognized as a peak, in NR3 format.
:SEARCh:PEAK:NPEaks <number> (see page 1191)	:SEARCh:PEAK:NPEaks? (see page 1191)	<number> ::= max number of peaks to find, 1-11 in NR1 format.
:SEARCh:PEAK:SOURce <source> (see page 1192)	:SEARCh:PEAK:SOURce? (see page 1192)	<source> ::= {FUNCTION<m> MATH<m> FFT} (must be an FFT waveform) <m> ::= 1 to (# math functions) in NR1 format
:SEARCh:PEAK:THRESHold <level> (see page 1193)	:SEARCh:PEAK:THRESHold? (see page 1193)	<level> ::= necessary level to be considered a peak, in NR3 format.

:SEARCh:PEAK:EXCursion

N (see [page 1708](#))

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 489
 - [":FUNCTION<m>\[:FFT\]:VTYPE"](#) on page 547
 - [":SEARCh:PEAK:NPEaks"](#) on page 1191
 - [":SEARCh:PEAK:SOURce"](#) on page 1192
 - [":SEARCh:PEAK:THReshold"](#) on page 1193

:SEARCh:PEAK:NPEaks

N (see [page 1708](#))

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 1190
 - [":SEARCh:PEAK:SOURce"](#) on page 1192
 - [":SEARCh:PEAK:THReshold"](#) on page 1193

:SEARCh:PEAK:SOURce

N (see [page 1708](#))

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 1190
 - [":SEARCh:PEAK:NPEaks"](#) on page 1191
 - [":SEARCh:PEAK:THReshold"](#) on page 1193

:SEARCh:PEAK:THReshold

N (see [page 1708](#))

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 489
 - [":FUNCTion<m>\[:FFT\]:VTYPE"](#) on page 547
 - [":SEARCh:PEAK:EXCursion"](#) on page 1190
 - [":SEARCh:PEAK:NPEaks"](#) on page 1191
 - [":SEARCh:PEAK:SOURce"](#) on page 1192

:SEARCh:RUNT Commands

Table 132 :SEARCh:RUNT Commands Summary

Command	Query	Options and Query Returns
:SEARCh:RUNT:POLarity <polarity> (see page 1195)	:SEARCh:RUNT:POLarity ? (see page 1195)	<polarity> ::= {POSitive NEGative EITHer}
:SEARCh:RUNT:QUALifie r <qualifier> (see page 1196)	:SEARCh:RUNT:QUALifie r? (see page 1196)	<qualifier> ::= {GREaterthan LESSthan NONE}
:SEARCh:RUNT:SOURce <source> (see page 1197)	:SEARCh:RUNT:SOURce? (see page 1197)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARCh:RUNT:TIME <time>[suffix] (see page 1198)	:SEARCh:RUNT:TIME? (see page 1198)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

:SEARCh:RUNT:POLarity

N (see [page 1708](#))

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 | EITH}

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:RUNT:SOURce"](#) on page 1197

:SEARCh:RUNT:QUALifier

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177

:SEARCh:RUNT:SOURce

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:RUNT:POLarity"](#) on page 1195

:SEARCh:RUNT:TIME

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:RUNT:QUALifier"](#) on page 1196

:SEARCh:TRANsition Commands

Table 133 :SEARCh:TRANsition Commands Summary

Command	Query	Options and Query Returns
:SEARCh:TRANsition:QU ALifier <qualifier> (see page 1200)	:SEARCh:TRANsition:QU ALifier? (see page 1200)	<qualifier> ::= {GREATERthan LESSthan}
:SEARCh:TRANsition:SL OPe <slope> (see page 1201)	:SEARCh:TRANsition:SL OPe? (see page 1201)	<slope> ::= {NEGative POSitive}
:SEARCh:TRANsition:SO URce <source> (see page 1202)	:SEARCh:TRANsition:SO URce? (see page 1202)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:SEARCh:TRANsition:TI ME <time>[suffix] (see page 1203)	:SEARCh:TRANsition:TI ME? (see page 1203)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

:SEARCh:TRANSition:QUALifier

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:TRANSition:TIME"](#) on page 1203

:SEARCh:TRANSition:SLOPe

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:TRANSition:SOURce"](#) on page 1202
 - [":SEARCh:TRANSition:TIME"](#) on page 1203

:SEARCh:TRANSition:SOURce

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:MODE"](#) on page 1177
 - [":SEARCh:TRANSition:SLOPe"](#) on page 1201

:SEARCh:TRANsition:TIME

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:TRANsition:QUALifier"](#) on page 1200

:SEARCh:SERial:A429 Commands

Table 134 :SEARCh:SERial:A429 Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:A429:LABel <value> (see page 1205)	:SEARCh:SERial:A429:LABel? (see page 1205)	<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:MODE <condition> (see page 1206)	:SEARCh:SERial:A429:MODE? (see page 1206)	<condition> ::= {LABel LBITs PERRor WERRor GERRor WGERrors ALLerrors}
:SEARCh:SERial:A429:PATTern:DATA <string> (see page 1207)	:SEARCh:SERial:A429:PATTern:DATA? (see page 1207)	<string> ::= "nn...n" where n ::= {0 1}, length depends on FORMat
:SEARCh:SERial:A429:PATTern:SDI <string> (see page 1208)	:SEARCh:SERial:A429:PATTern:SDI? (see page 1208)	<string> ::= "nn" where n ::= {0 1}, length always 2 bits
:SEARCh:SERial:A429:PATTern:SSM <string> (see page 1209)	:SEARCh:SERial:A429:PATTern:SSM? (see page 1209)	<string> ::= "nn" where n ::= {0 1}, length always 2 bits

:SEARCh:SERial:A429:LABel

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479

• [":SEARCh:SERial:A429:MODE"](#) on page 1206

:SEARCh:SERial:A429:MODE

N (see [page 1708](#))

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 1663

- See Also**
- ["Introduction to :SBUS<n> Commands"](#) on page 983
 - [":SBUS<n>:MODE"](#) on page 987
 - [":SEARCh:SERial:A429:LABel"](#) on page 1205
 - [":SEARCh:SERial:A429:PATtern:DATA"](#) on page 1207
 - [":SEARCh:SERial:A429:PATtern:SDI"](#) on page 1208
 - [":SEARCh:SERial:A429:PATtern:SSM"](#) on page 1209
 - [":SBUS<n>:A429:SOURce"](#) on page 995

:SEARCh:SERIal:A429:PATTerN:DATA

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SEARCh:SERIal:A429:MODE"](#) on page 1206
 • [":SEARCh:SERIal:A429:PATTerN:SDI"](#) on page 1208
 • [":SEARCh:SERIal:A429:PATTerN:SSM"](#) on page 1209

:SEARCh:SERial:A429:PATtern:SDI

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:A429:FORMat"](#) on page 993
 • [":SEARCh:SERial:A429:MODE"](#) on page 1206
 • [":SEARCh:SERial:A429:PATtern:DATA"](#) on page 1207
 • [":SEARCh:SERial:A429:PATtern:SSM"](#) on page 1209

:SEARCh:SERial:A429:PATtern:SSM

N (see [page 1708](#))

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 1663

See Also • ["Introduction to :TRIGger Commands"](#) on page 1479
 • [":SBUS<n>:A429:FORMat"](#) on page 993
 • [":SEARCh:SERial:A429:MODE"](#) on page 1206
 • [":SEARCh:SERial:A429:PATtern:DATA"](#) on page 1207
 • [":SEARCh:SERial:A429:PATtern:SDI"](#) on page 1208

:SEARCH:SERIAL:CAN Commands

Table 135 :SEARCH:SERIAL:CAN Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:CAN:MODE <value> (see page 1211)	:SEARCH:SERIAL:CAN:MODE? (see page 1211)	<value> ::= {IDEither IDData DATA IDRemote ERROR ACKerror FORMerror STUFFerror CRCerror ALLerrors OVERload MESSAGE MSIGNAL}
:SEARCH:SERIAL:CAN:PA TTern:DATA <string> (see page 1213)	:SEARCH:SERIAL:CAN:PA TTern:DATA? (see page 1213)	<string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X} for hexadecimal
:SEARCH:SERIAL:CAN:PA TTern:DATA:LENGTH <length> (see page 1214)	:SEARCH:SERIAL:CAN:PA TTern:DATA:LENGTH? (see page 1214)	<length> ::= integer from 1 to 8 in NR1 format
:SEARCH:SERIAL:CAN:PA TTern:ID <string> (see page 1215)	:SEARCH:SERIAL:CAN:PA TTern:ID? (see page 1215)	<string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X} for hexadecimal
:SEARCH:SERIAL:CAN:PA TTern:ID:MODE <value> (see page 1216)	:SEARCH:SERIAL:CAN:PA TTern:ID:MODE? (see page 1216)	<value> ::= {STANDARD EXTENDED}
:SEARCH:SERIAL:CAN:SY MBolic:MESSAGE <name> (see page 1217)	:SEARCH:SERIAL:CAN:SY MBolic:MESSAGE? (see page 1217)	<name> ::= quoted ASCII string
:SEARCH:SERIAL:CAN:SY MBolic:SIGNAL <name> (see page 1218)	:SEARCH:SERIAL:CAN:SY MBolic:SIGNAL? (see page 1218)	<name> ::= quoted ASCII string
:SEARCH:SERIAL:CAN:SY MBolic:VALUE <data> (see page 1219)	:SEARCH:SERIAL:CAN:SY MBolic:VALUE? (see page 1219)	<data> ::= value in NR3 format

:SEARCh:SERial:CAN:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:CAN:MODE <value>

```
<value> ::= {IDEither | IDData | DATA | IDRemote | ERRor | ACKerror
            | FORMerror | STUFferror | CRCerror | ALLerrors | OVERload
            | MESSage | MSIGnal}
```

The :SEARCh:SERial:CAN:MODE command selects the type of CAN information to find in the Lister display:

Condition	Front-panel name	Description
IDEither	Frame ID	Finds remote or data frames matching the specified ID.
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.
ERRor	Error Frame	Finds CAN active error frames.
ACKerror	Acknowledge Error	Finds the acknowledge bit if the polarity is incorrect.
FORMerror	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 Field 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> ::= {IDE | IDD | DATA | IDR | ERR | ACK | FORM | STUF | CRC
            | ALL | OVER | MESS | MSIG}
```

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:CAN:PATtern:DATA"](#) on page 1213
 - [":SEARCh:SERial:CAN:PATtern:ID"](#) on page 1215

- **":RECall:DBC[:START]"** on page 945
- **":SEARch:SERial:CAN:SYMBolic:MESSage"** on page 1217
- **":SEARch:SERial:CAN:SYMBolic:SIGNal"** on page 1218
- **":SEARch:SERial:CAN:SYMBolic:VALue"** on page 1219

:SEARCh:SERIal:CAN:PATTerN:DATA

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERIal:CAN:PATTerN:DATA:LENGth"](#) on page 1214

:SEARCh:SERial:CAN:PATtern:DATA:LENGth

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERial:CAN:PATtern:DATA"](#) on page 1213

:SEARCh:SERIal:CAN:PATtern:ID

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERIal:CAN:PATtern:ID:MODE"](#) on page 1216

:SEARCh:SERIal:CAN:PATTErn:ID:MODE

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERIal:CAN:PATTErn:ID"](#) on page 1215

:SEARCh:SERIal:CAN:SYMBolic:MESSage

N (see [page 1708](#))

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 945
 - [":SEARCh:SERIal:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERIal:CAN:SYMBolic:SIGNaI"](#) on page 1218
 - [":SEARCh:SERIal:CAN:SYMBolic:VALue"](#) on page 1219

:SEARCh:SERIal:CAN:SYMBolic:SIGNal

N (see [page 1708](#))

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 945
 - [":SEARCh:SERIal:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERIal:CAN:SYMBolic:MESSage"](#) on page 1217
 - [":SEARCh:SERIal:CAN:SYMBolic:VALue"](#) on page 1219

:SEARCh:SERial:CAN:SYMBolic:VALue

N (see [page 1708](#))

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 945
 - [":SEARCh:SERial:CAN:MODE"](#) on page 1211
 - [":SEARCh:SERial:CAN:SYMBolic:MESSAge"](#) on page 1217
 - [":SEARCh:SERial:CAN:SYMBolic:SIGNal"](#) on page 1218

:SEARCh:SERial:IIC Commands

Table 136 :SEARCh:SERial:IIC Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:IIC:MODE <value> (see page 1221)	:SEARCh:SERial:IIC:MODE? (see page 1221)	<value> ::= {REStart ADDRESS ANACK NACKnowledge REPRom READ7 WRITE7 R7Data2 W7Data2}
:SEARCh:SERial:IIC:PA TTern:ADDRESS <value> (see page 1223)	:SEARCh:SERial:IIC:PA TTern:ADDRESS? (see page 1223)	<value> ::= integer
:SEARCh:SERial:IIC:PA TTern:DATA <value> (see page 1224)	:SEARCh:SERial:IIC:PA TTern:DATA? (see page 1224)	<value> ::= integer
:SEARCh:SERial:IIC:PA TTern:DATA2 <value> (see page 1225)	:SEARCh:SERial:IIC:PA TTern:DATA2? (see page 1225)	<value> ::= integer
:SEARCh:SERial:IIC:QU ALifier <value> (see page 1226)	:SEARCh:SERial:IIC:QU ALifier? (see page 1226)	<value> ::= {EQUal NOTequal LESSthan GREaterthan}

:SEARCh:SERial:IIC:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:IIC:MODE <value>

```
<value> ::= {REStart | ADDRESS | ANACk | NACKnowledge | REPRom
             | READ7 | WRITe7 | R7Data2 | W7Data2}
```

The :SEARCh:SERial:IIC:MODE command selects the type of IIC information to find in the Lister display:

- REStart – searches for another start condition occurring before a stop condition.
- ADDRESS – searches for a packet with the specified address, ignoring the R/W bit.
- ANACk – searches for address with no acknowledge events.
- NACKnowledge – searches for missing acknowledge events.
- REPRom – searches for EEPROM data reads.
- 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.
- 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.

NOTE

The short form of READ7 (READ7), REPRom (REPR), and WRITe7 (WRIT7) do not follow the defined Long Form to Short Form Truncation Rules (see [page 1710](#)).

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> ::= {REST | ADDR | ANAC | NACK | REPR | READ7 | WRITe7
             | R7D2 | W7D2}
```

- See Also**
- **Chapter 36**, “:SEARCh Commands,” starting on page 1173
 - **“:SEARCh:SERial:IIC:PATtern:ADDReSS”** on page 1223
 - **“:SEARCh:SERial:IIC:PATtern:DATA”** on page 1224
 - **“:SEARCh:SERial:IIC:PATtern:DATA2”** on page 1225
 - **“:SEARCh:SERial:IIC:QUALifier”** on page 1226

:SEARCh:SERIal:IIC:PATTeRn:ADDReSS

N (see [page 1708](#))

Command Syntax :SEARCh:SERIal:IIC:PATTeRn:ADDReSS <value>

<value> ::= integer

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:IIC:MODE"](#) on page 1221

:SEARCh:SERial:IIC:PATtern:DATA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:IIC:PATtern:DATA <value>

<value> ::= integer

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:IIC:MODE"](#) on page 1221
 - [":SEARCh:SERial:IIC:QUALifier"](#) on page 1226
 - [":SEARCh:SERial:IIC:PATtern:DATA2"](#) on page 1225

:SEARCh:SERial:IIC:PATtern:DATA2

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:IIC:PATtern:DATA2 <value>

<value> ::= integer

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:IIC:MODE"](#) on page 1221
 - [":SEARCh:SERial:IIC:PATtern:DATA"](#) on page 1224

:SEARCh:SERial:IIC:QUALifier

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:IIC:MODE"](#) on page 1221
 - [":SEARCh:SERial:IIC:PATtern:DATA"](#) on page 1224

:SEARCh:SERial:LIN Commands

Table 137 :SEARCh:SERial:LIN Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:LIN:ID <value> (see page 1228)	:SEARCh:SERial:LIN:ID ? (see page 1228)	<value> ::= 7-bit integer in decimal or <nondecimal> from 0-63 <nondecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 1} for binary
:SEARCh:SERial:LIN:MO DE <value> (see page 1229)	:SEARCh:SERial:LIN:MO DE? (see page 1229)	<value> ::= {ID DATA ERRor}
:SEARCh:SERial:LIN:PA TTern:DATA <string> (see page 1230)	:SEARCh:SERial:LIN:PA TTern:DATA? (see page 1230)	When :SEARCh:SERial:LIN:PA TTern:FORMa t DECimal, <string> ::= "n" where n ::= 32-bit integer in unsigned decimal, returns "\$" if data has any don't cares When :SEARCh:SERial:LIN:PA TTern:FORMa t HEX, <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X }
:SEARCh:SERial:LIN:PA TTern:DATA:LENGth <length> (see page 1231)	:SEARCh:SERial:LIN:PA TTern:DATA:LENGth? (see page 1231)	<length> ::= integer from 1 to 8 in NR1 format
:SEARCh:SERial:LIN:PA TTern:FORMat <base> (see page 1232)	:SEARCh:SERial:LIN:PA TTern:FORMat? (see page 1232)	<base> ::= {HEX DECimal}
:SEARCh:SERial:LIN:SY MBolic:FRAME <name> (see page 1233)	:SEARCh:SERial:LIN:SY MBolic:FRAME? (see page 1233)	<name> ::= quoted ASCII string
:SEARCh:SERial:LIN:SY MBolic:SIGNAL <name> (see page 1234)	:SEARCh:SERial:LIN:SY MBolic:SIGNAL? (see page 1234)	<name> ::= quoted ASCII string
:SEARCh:SERial:LIN:SY MBolic:VALue <data> (see page 1235)	:SEARCh:SERial:LIN:SY MBolic:VALue? (see page 1235)	<data> ::= value in NR3 format

:SEARCh:SERIal:LIN:ID

N (see [page 1708](#))

Command Syntax :SEARCh:SERIal:LIN:ID <value>

<value> ::= 7-bit integer in decimal or <nondecimal> from 0-63

<nondecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

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

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:LIN:MODE"](#) on page 1229

:SEARCh:SERial:LIN:MODE

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:LIN:ID"](#) on page 1228
 - [":SEARCh:SERial:LIN:PATtern:DATA"](#) on page 1230
 - [":RECall:LDF\[:STARt\]"](#) on page 947
 - [":SEARCh:SERial:LIN:SYMBolic:FRAMe"](#) on page 1233
 - [":SEARCh:SERial:LIN:SYMBolic:SIGNaL"](#) on page 1234
 - [":SEARCh:SERial:LIN:SYMBolic:VALue"](#) on page 1235

:SEARCH:SERIAL:LIN:PATTERN:DATA

N (see [page 1708](#))

Command Syntax :SEARCH:SERIAL:LIN:PATTERN:DATA <string>

When :SEARCH:SERIAL:LIN:PATTERN:FORMAT DECIMAL,
<string> ::= "n" where n ::= 32-bit integer in unsigned decimal

When :SEARCH:SERIAL:LIN:PATTERN:FORMAT HEX,
<string> ::= "0xnn...n" 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.

Query Syntax :SEARCH:SERIAL:LIN:PATTERN:DATA?

The :SEARCH:SERIAL:LIN:PATTERN:DATA? query returns the current data value setting.

Return Format <string><NL>

When :SEARCH:SERIAL:LIN:PATTERN:FORMAT DECIMAL,
<string> ::= "n" where n ::= 32-bit integer in unsigned decimal or
"\$" if data has any don't cares

When :SEARCH:SERIAL:LIN:PATTERN:FORMAT HEX,
<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X }

- See Also**
- [Chapter 36](#), “:SEARCH Commands,” starting on page 1173
 - [":SEARCH:SERIAL:LIN:MODE"](#) on page 1229
 - [":SEARCH:SERIAL:LIN:PATTERN:FORMAT"](#) on page 1232
 - [":SEARCH:SERIAL:LIN:PATTERN:DATA:LENGTH"](#) on page 1231

:SEARCh:SERIal:LIN:PATTerN:DATA:LENGTh

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:LIN:PATTerN:DATA"](#) on page 1230

:SEARCh:SERIal:LIN:PATtern:FORMat

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:LIN:PATtern:DATA"](#) on page 1230

:SEARCh:SERial:LIN:SYMBolic:FRAME

N (see [page 1708](#))

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 947
 - [":SEARCh:SERial:LIN:MODE"](#) on page 1229
 - [":SEARCh:SERial:LIN:SYMBolic:SIGNal"](#) on page 1234
 - [":SEARCh:SERial:LIN:SYMBolic:VALue"](#) on page 1235

:SEARCh:SERial:LIN:SYMBolic:SIGNal

N (see [page 1708](#))

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 947
 - [":SEARCh:SERial:LIN:MODE"](#) on page 1229
 - [":SEARCh:SERial:LIN:SYMBolic:FRAME"](#) on page 1233
 - [":SEARCh:SERial:LIN:SYMBolic:VALue"](#) on page 1235

:SEARCh:SERial:LIN:SYMBolic:VALue

N (see [page 1708](#))

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 947
 - [":SEARCh:SERial:LIN:MODE"](#) on page 1229
 - [":SEARCh:SERial:LIN:SYMBolic:FRAMe"](#) on page 1233
 - [":SEARCh:SERial:LIN:SYMBolic:SIGNal"](#) on page 1234

:SEARCh:SERial:M1553 Commands

Table 138 :SEARCh:SERial:M1553 Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:M1553:MODE <value> (see page 1237)	:SEARCh:SERial:M1553:MODE? (see page 1237)	<value> ::= {DStart CStart RTA RTA11 PERRor SERRor MERRor}
:SEARCh:SERial:M1553:PATtern:DATA <string> (see page 1238)	:SEARCh:SERial:M1553:PATtern:DATA? (see page 1238)	<string> ::= "nn...n" where n ::= {0 1}
:SEARCh:SERial:M1553:RTA <value> (see page 1239)	:SEARCh:SERial:M1553:RTA? (see page 1239)	<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 1708](#))

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> ::= {DST | CST | RTA | RTA11 | PERR | SERR | MERR}

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:M1553:RTA"](#) on page 1239
 - [":SEARCh:SERial:M1553:PATtern:DATA"](#) on page 1238

:SEARCh:SERial:M1553:PATtern:DATA

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:M1553:MODE"](#) on page 1237

:SEARCh:SERial:M1553:RTA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:M1553:RTA <value>

<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}

The :SEARCh:SERial:M1553:RTA command specifies the Remote Terminal Address (RTA) value when searching for MIL-STD-1553 events.

Query Syntax :SEARCh:SERial:M1553:RTA?

The :SEARCh:SERial:M1553:RTA? query returns the current Remote Terminal Address value setting.

Return Format <value><NL>

<value> ::= 5-bit integer in decimal from 0-31

See Also • [Chapter 36](#), “:SEARCh Commands,” starting on page 1173

:SEARCh:SERial:SENT Commands

Table 139 :SEARCh:SERial:SENT Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:SENT:F AST:DATA <string> (see page 1241)	:SEARCh:SERial:SENT:F AST:DATA? (see page 1241)	<string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SEARCh:SERial:SENT:M ODE <mode> (see page 1242)	:SEARCh:SERial:SENT:M ODE? (see page 1242)	<mode> ::= {FCData SCMid SCData CRCerror PPErrors}
:SEARCh:SERial:SENT:S LOW:DATA <data> (see page 1243)	:SEARCh:SERial:SENT:S LOW:DATA? (see page 1243)	<data> ::= from -1 (don't care) to 65535, in NR1 format.
:SEARCh:SERial:SENT:S LOW:ID <id> (see page 1244)	:SEARCh:SERial:SENT:S LOW:ID? (see page 1244)	<id> ::= from -1 (don't care) to 255, in NR1 format.

:SEARCh:SERial:SENT:FAST:DATA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:SENT:FAST:DATA <string>

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

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.

Query Syntax :SEARCh:SERial:SENT:FAST:DATA?

The :SEARCh:SERial:SENT:FAST:DATA? query returns the fast channel data search value setting.

Return Format <string><NL>

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

- See Also**
- [":SEARCh:SERial:SENT:MODE"](#) on page 1242
 - [":SEARCh:SERial:SENT:SLOW:DATA"](#) on page 1243
 - [":SEARCh:SERial:SENT:SLOW:ID"](#) on page 1244

:SEARCH:SERIAL:SENT:MODE

N (see [page 1708](#))

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.
- SCMid – finds Slow Channel Message IDs that match the value entered.
- SCData – finds Slow Channel Message IDs and Data that match the values entered.
- 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 1241
 - [":SEARCH:SERIAL:SENT:SLOW:DATA"](#) on page 1243
 - [":SEARCH:SERIAL:SENT:SLOW:ID"](#) on page 1244

:SEARCh:SErIal:SENT:SLOW:DATA

N (see [page 1708](#))

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 1241
 - [":SEARCh:SErIal:SENT:MODE"](#) on page 1242
 - [":SEARCh:SErIal:SENT:SLOW:ID"](#) on page 1244

:SEARCh:SERial:SENT:SLOW:ID

N (see [page 1708](#))

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 1241
 - [":SEARCh:SERial:SENT:MODE"](#) on page 1242
 - [":SEARCh:SERial:SENT:SLOW:DATA"](#) on page 1243

:SEARCh:SERial:SPI Commands

Table 140 :SEARCh:SERial:SPI Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:SPI:MODE <value> (see page 1246)	:SEARCh:SERial:SPI:MODE? (see page 1246)	<value> ::= {MOSI MISO}
:SEARCh:SERial:SPI:PA TTern:DATA <string> (see page 1247)	:SEARCh:SERial:SPI:PA TTern:DATA? (see page 1247)	<string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X}
:SEARCh:SERial:SPI:PA TTern:WIDTh <width> (see page 1248)	:SEARCh:SERial:SPI:PA TTern:WIDTh? (see page 1248)	<width> ::= integer from 1 to 10

:SEARCh:SERial:SPI:MODE

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:SPI:PATtern:DATA"](#) on page 1247

:SEARCh:SERial:SPI:PATtern:DATA

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:SPI:PATtern:WIDTh"](#) on page 1248

:SEARCh:SERIal:SPI:PATtern:WIDTh

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:SPI:PATtern:DATA"](#) on page 1247

:SEARCh:SERial:UART Commands

Table 141 :SEARCh:SERial:UART Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:UART:D ATA <value> (see page 1250)	:SEARCh:SERial:UART:D ATA? (see page 1250)	<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal, <hexadecimal>, or <binary> format <hexadecimal> ::= #Hnn where n ::= {0,...,9 A,...,F} for hexadecimal <binary> ::= #Bnn...n where n ::= {0 1} for binary
:SEARCh:SERial:UART:M ODE <value> (see page 1251)	:SEARCh:SERial:UART:M ODE? (see page 1251)	<value> ::= {RDATa RD1 RD0 RDX TDATAa TD1 TD0 TDX RXPARity TXPARity AERRor}
:SEARCh:SERial:UART:Q UALifier <value> (see page 1253)	:SEARCh:SERial:UART:Q UALifier? (see page 1253)	<value> ::= {EQUal NOTequal GREaterthan LESSthan}

:SEARCh:SERial:UART:DATA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:UART:DATA <value>

<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal,
<hexadecimal>, or <binary> format

<hexadecimal> ::= #Hnn where n ::= {0,...,9 | A,...,F} for hexadecimal

<binary> ::= #Bnn...n where n ::= {0 | 1} for binary

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.

Query Syntax :SEARCh:SERial:UART:DATA?

The :SEARCh:SERial:UART:DATA? query returns the current data value setting.

Return Format <value><NL>

<value> ::= 8-bit integer from 0-255 (0x00-0xff) in decimal format

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:UART:MODE"](#) on page 1251
 - [":SEARCh:SERial:UART:QUALifier"](#) on page 1253

:SEARCh:SERial:UART:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:UART:MODE <value>

```
<value> ::= {RDATa | RD1 | RD0 | RDX | TDATa | TD1 | TD0 | TDX
            | RXPArity | TXPArity | 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).
- RXPArity – searches for receive data parity errors.
- TXPArity – searches for transmit data 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 | RXPA
            | TXPA | AERR}
```

- See Also**
- [Chapter 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERial:UART:DATA"](#) on page 1250

- **":SEARCh:SERIal:UART:QUALifier"** on page 1253

:SEARCh:SERIal:UART:QUALifier

N (see [page 1708](#))

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 36](#), “:SEARCh Commands,” starting on page 1173
 - [":SEARCh:SERIal:UART:DATA"](#) on page 1250

:SEARCH:SERIAL:USB Commands

Table 142 :SEARCH:SERIAL:USB Commands Summary

Command	Query	Options and Query Returns
:SEARCH:SERIAL:USB:MODE <condition> (see page 1256)	:SEARCH:SERIAL:USB:MODE? (see page 1256)	<event> ::= {Token DATA HANDshake SPECial ALLerrors PIDerror CRC5error CRC16error GLITCherror STUFFerror SE1error}
:SEARCH:SERIAL:USB:ADDRESS <string> (see page 1257)	:SEARCH:SERIAL:USB:ADDRESS? (see page 1257)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:CRCC <string> (see page 1258)	:SEARCH:SERIAL:USB:CRCC? (see page 1258)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:DATA <string> (see page 1259)	:SEARCH:SERIAL:USB:DATA? (see page 1259)	<string> ::= "0xn..." where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:DATA:LENGTH <value> (see page 1260)	:SEARCH:SERIAL:USB:DATA:LENGTH? (see page 1260)	<length> ::= data length between 1-20
:SEARCH:SERIAL:USB:ENDPOINT <string> (see page 1261)	:SEARCH:SERIAL:USB:ENDPOINT? (see page 1261)	<string> ::= "0xn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:ET <string> (see page 1262)	:SEARCH:SERIAL:USB:ET? (see page 1262)	<string> ::= "0xn" where n ::= {0 1 2 3 X \$}
:SEARCH:SERIAL:USB:FRAME <string> (see page 1263)	:SEARCH:SERIAL:USB:FRAME? (see page 1263)	<string> ::= "0xnnnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:HANDADDRESS <string> (see page 1264)	:SEARCH:SERIAL:USB:HANDADDRESS? (see page 1264)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCH:SERIAL:USB:PID:DATA <pid> (see page 1265)	:SEARCH:SERIAL:USB:PID:DATA? (see page 1265)	<pid> ::= {DATA0 DATA1 DATA2 MDATA}
:SEARCH:SERIAL:USB:PID:HANDshake <pid> (see page 1266)	:SEARCH:SERIAL:USB:PID:HANDshake? (see page 1266)	<pid> ::= {ACK NAK STALL NYET}
:SEARCH:SERIAL:USB:PID:SPECIAL <pid> (see page 1267)	:SEARCH:SERIAL:USB:PID:SPECIAL? (see page 1267)	<pid> ::= {PING PRE ERR SPLit}

Table 142 :SEARCh:SERIal:USB Commands Summary (continued)

Command	Query	Options and Query Returns
:SEARCh:SERIal:USB:PI D:TOKE n <pid> (see page 1268)	:SEARCh:SERIal:USB:PI D:TOKE n? (see page 1268)	<pid> ::= {OUT IN SETup SOF}
:SEARCh:SERIal:USB:PO RT <string> (see page 1269)	:SEARCh:SERIal:USB:PO RT? (see page 1269)	<string> ::= "0xnn" where n ::= {0,...,9 A,...,F X \$}
:SEARCh:SERIal:USB:SC <string> (see page 1270)	:SEARCh:SERIal:USB:SC ? (see page 1270)	<string> ::= "0xn" where n ::= {0 1 X \$}
:SEARCh:SERIal:USB:SE U <string> (see page 1271)	:SEARCh:SERIal:USB:SE U? (see page 1271)	<string> ::= "0xn" where n ::= {0 1 2 3 X \$}

:SEARCh:SERial:USB:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:MODE <mode>

```
<mode> ::= {TOKen | DATA | HANDshake | SPECial | ALLerrors  
            | PIDerror | CRC5error | CRC16error | GLITCherror  
            | STUFFerror | SE1error}
```

The :SEARCh:SERial:USB:MODE command specifies the USB search mode.

Query Syntax :SEARCh:SERial:USB:MODE?

The :SEARCh:SERial:USB:MODE? query returns the specified USB search mode.

Return Format <mode><NL>

```
<mode> ::= {TOK | DATA | HAND | SPEC | ALL | PID | CRC5  
            | CRC16 | GLIT | STUFF | SE1}
```

- See Also**
- [":SEARCh:SERial:USB:PID:DATA"](#) on page 1265
 - [":SEARCh:SERial:USB:PID:HANDshake"](#) on page 1266
 - [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267
 - [":SEARCh:SERial:USB:PID:TOKen"](#) on page 1268

:SEARCh:SERial:USB:ADDResS

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:ADDResS <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:ADDResS command specifies the 7-bit Address portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:ADDResS?

The :SEARCh:SERial:USB:ADDResS? query returns the specified Address portion of the search value.

Return Format <string><NL>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

- See Also**
- [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267
 - [":SEARCh:SERial:USB:PID:TOKen"](#) on page 1268

:SEARCh:SERial:USB:CRC

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:CRC <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:CRC command specifies the CRC portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:CRC?

The :SEARCh:SERial:USB:CRC? query returns the specified CRC portion of the search value.

Return Format <string><NL>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

- See Also**
- [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267
 - [":SEARCh:SERial:USB:PID:TOKen"](#) on page 1268

:SEARCh:SERial:USB:DATA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:DATA <string>

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:DATA command specifies the Data portion of the search value, in hex.

See the :SEARCh:SERial:USB:DATA:LENGth command for setting the length of the data value.

Query Syntax :SEARCh:SERial:USB:DATA?

The :SEARCh:SERial:USB:DATA? query returns the specified Data portion of the search value.

Return Format <string><NL>

<string> ::= "0xn..." where n ::= {0,...,9 | A,...,F | X | \$}

See Also • [":SEARCh:SERial:USB:DATA:LENGth"](#) on page 1260

:SEARCh:SERial:USB:DATA:LENGth

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:DATA:LENGth <length>

<length> ::= data length between 1-20

The :SEARCh:SERial:USB:DATA:LENGth command specifies the data length in bytes.

Query Syntax :SEARCh:SERial:USB:DATA:LENGth?

The :SEARCh:SERial:USB:DATA:LENGth? query returns the specified data length.

Return Format <length><NL>

<length> ::= data length between 1-20

See Also • [":SEARCh:SERial:USB:DATA"](#) on page 1259

:SEARCh:SERial:USB:ENDPoint

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:ENDPoint <string>

<string> ::= "0xn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:ENDPoint command specifies the 4-bit Endpoint portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:ENDPoint?

The :SEARCh:SERial:USB:ENDPoint? query returns the specified Endpoint portion of the search value.

Return Format <string><NL>

<string> ::= "0xn" where n ::= {0,...,9 | A,...,F | X | \$}

- See Also**
- [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267
 - [":SEARCh:SERial:USB:PID:TOKen"](#) on page 1268

:SEARCh:SERial:USB:ET

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:ET <string>

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

The :SEARCh:SERial:USB:ET command specifies the 2-bit ET portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:ET?

The :SEARCh:SERial:USB:ET? query returns the specified ET portion of the search value.

Return Format <string><NL>

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

See Also • [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267

:SEARCh:SERial:USB:FRAMe

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:FRAMe <string>

<string> ::= "0xnnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:FRAMe command specifies the 11-bit Frame portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:FRAMe?

The :SEARCh:SERial:USB:FRAMe? query returns the specified Frame portion of the search value.

Return Format <string><NL>

<string> ::= "0xnnn" where n ::= {0,...,9 | A,...,F | X | \$}

See Also • [":SEARCh:SERial:USB:PID:TOKE n"](#) on page 1268

:SEARCh:SERial:USB:HADDress

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:HADDress <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:HADDress command specifies the 7-bit Hub Address portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:HADDress?

The :SEARCh:SERial:USB:HADDress? query returns the specified Hub Address portion of the search value.

Return Format <string><NL>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

See Also • [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267

:SEARCh:SERial:USB:PID:DATA

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:PID:DATA <pid>

<pid> ::= {DATA0 | DATA1 | DATA2 | MDATA}

The :SEARCh:SERial:USB:PID:DATA command specifies the USB data PID to find.

The DATA USB search mode can be selected using the :SEARCh:SERial:USB:MODE command.

Query Syntax :SEARCh:SERial:USB:PID:DATA?

The :SEARCh:SERial:USB:PID:DATA? query returns the specified data PID.

Return Format <pid><NL>

<pid> ::= {DATA0 | DATA1 | DATA2 | MDATA}

See Also • [":SEARCh:SERial:USB:MODE"](#) on page 1256

:SEARCh:SERial:USB:PID:HANDshake

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:PID:HANDshake <pid>

<pid> ::= {ACK | NAK | STAl1 | NYET}

The :SEARCh:SERial:USB:PID:HANDshake command specifies the USB handshake PID to find.

The HANDshake USB search mode can be selected using the :SEARCh:SERial:USB:MODE command.

Query Syntax :SEARCh:SERial:USB:PID:HANDshake?

The :SEARCh:SERial:USB:PID:HANDshake? query returns the specified handshake PID.

Return Format <pid><NL>

<pid> ::= {ACK | NAK | STAl | NYET}

See Also • [":SEARCh:SERial:USB:MODE"](#) on page 1256

:SEARCh:SERial:USB:PID:SPECial

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:PID:SPECial <pid>

<pid> ::= {PING | PRE | ERR | SPLit}

The :SEARCh:SERial:USB:PID:SPECial command specifies the USB special PID to find.

The SPECial USB search mode can be selected using the :SEARCh:SERial:USB:MODE command.

Query Syntax :SEARCh:SERial:USB:PID:SPECial?

The :SEARCh:SERial:USB:PID:SPECial? query returns the specified special PID.

Return Format <pid><NL>

<pid> ::= {PING | PRE | ERR | SPL}

See Also • [":SEARCh:SERial:USB:MODE"](#) on page 1256

:SEARCh:SERial:USB:PID:TOKen

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:PID:TOKen <pid>

<pid> ::= {OUT | IN | SETup | SOF}

The :SEARCh:SERial:USB:PID:TOKen command specifies the USB token PID to find.

The TOKen USB search mode can be selected using the :SEARCh:SERial:USB:MODE command.

Query Syntax :SEARCh:SERial:USB:PID:TOKen?

The :SEARCh:SERial:USB:PID:TOKen? query returns the specified token PID.

Return Format <pid><NL>

<pid> ::= {OUT | IN | SETup | SOF}

See Also • [":SEARCh:SERial:USB:MODE"](#) on page 1256

:SEARCh:SERial:USB:PORT

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:PORT <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

The :SEARCh:SERial:USB:PORT command specifies the 7-bit Port portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:PORT?

The :SEARCh:SERial:USB:PORT? query returns the specified Port portion of the search value.

Return Format <string><NL>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F | X | \$}

See Also • [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267

:SEARCh:SERial:USB:SC

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:SC <string>

<string> ::= "0xn" where n ::= {0 | 1 | X | \$}

The :SEARCh:SERial:USB:SC command specifies the 1-bit SC portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:SC?

The :SEARCh:SERial:USB:SC? query returns the specified SC portion of the search value.

Return Format <string><NL>

<string> ::= "0xn" where n ::= {0 | 1 | X | \$}

See Also • [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267

:SEARCh:SERial:USB:SEU

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USB:SEU <string>

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

The :SEARCh:SERial:USB:SEU command specifies the 2-bit S and E or U portion of the search value, in hex.

Query Syntax :SEARCh:SERial:USB:SEU?

The :SEARCh:SERial:USB:SEU? query returns the specified S and E or U portion of the search value.

Return Format <string><NL>

<string> ::= "0xn" where n ::= {0 | 1 | 2 | 3 | X | \$}

See Also • [":SEARCh:SERial:USB:PID:SPECial"](#) on page 1267

:SEARCh:SERial:USBPd Commands

Table 143 :SEARCh:SERial:USBPd Commands Summary

Command	Query	Options and Query Returns
:SEARCh:SERial:USBPd:MODE <mode> (see page 1273)	:SEARCh:SERial:USBPd:MODE? (see page 1273)	<mode> ::= {SOP SPrime SDPrime SPDebug SDPDebug HRST CRST CRCerror PERRor HEADer}
:SEARCh:SERial:USBPd:HEADer <type> (see page 1274)	:SEARCh:SERial:USBPd:HEADer? (see page 1274)	<type> ::= {CMESsage DMESsage EMESsage VALue}
:SEARCh:SERial:USBPd:HEADer:CMESsage <type> (see page 1276)	:SEARCh:SERial:USBPd:HEADer:CMESsage? (see page 1276)	<type> ::= {GOODcrc GOTomin ACcept REJect PING PSRDy GSRCap GSNCap DRSSwap PRSSwap VCSwap WAIT SRST GSCX GSStatus FRSSwap GPStatus GCCodes}
:SEARCh:SERial:USBPd:HEADer:DMESsage <type> (see page 1278)	:SEARCh:SERial:USBPd:HEADer:DMESsage? (see page 1278)	<type> ::= {SRCap REQuest BIST SNCap BStatus ALERt GCInfo VDEfined}
:SEARCh:SERial:USBPd:HEADer:EMESsage <type> (see page 1279)	:SEARCh:SERial:USBPd:HEADer:EMESsage? (see page 1279)	<type> ::= {SCX STATus GBCap GBStatus BCAP GMINfo MINfo SREQuest SRESponse FREQuest FRESponse PStatus CINfo CCODEs}
:SEARCh:SERial:USBPd:HEADer:QUALifier <type> (see page 1281)	:SEARCh:SERial:USBPd:HEADer:QUALifier? (see page 1281)	<type> ::= {NONE SOP SPrime SDPrime}
:SEARCh:SERial:USBPd:HEADer:VALue <string> (see page 1282)	:SEARCh:SERial:USBPd:HEADer:VALue? (see page 1282)	<string> ::= "nn...n" where n ::= {0 1 X} <string> ::= "0xnn...n" where n ::= {0,...,9 A,...,F X \$}

:SEARCh:SERial:USBPd:MODE

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:MODE <mode>

<mode> ::= {SOP | SPRime | SDPRime | SPDeBug | SDPDeBug | HRST | CRST
| CRCError | PERRor | HEADer}

The :SEARCh:SERial:USBPd:MODE command selects the USB PD search mode:

- SOP – SOP, searches ordered set Sync-1, Sync-1, Sync-1, Sync-2.
- SPRime – SOP', searches ordered set Sync-1, Sync-1, Sync-3, Sync-3.
- SDPRime – SOP", searches ordered set Sync-1, Sync-3, Sync-1, Sync-3.
- SPDeBug – SOP' Debug, searches ordered set Sync-1, RST-2, RST-2, Sync-3.
- SDPDeBug – SOP" Debug, searches ordered set Sync-1, RST-2, Sync-3, Sync-2.
- HRST – Hard Reset, searches ordered set RST-1, RST-1, RST-1, RST-2.
- CRST – Cable Reset, searches ordered set RST-1, Sync-1, RST-1, Sync-3.
- CRCError – CRC Error, searches when an error is detected on a 32-bit CRC.
- PERRor – Preamble Error, searches when an error is detected on a 64-bit sequence of alternating 0 and 1.
- HEADer – Header Content, searches a user-defined 16-bit value.

Query Syntax :SEARCh:SERial:USBPd:MODE?

The :SEARCh:SERial:USBPd:MODE? query returns the selected USB PD search mode.

Return Format <mode><NL>

<mode> ::= {SOP | SPR | SDPR | SPD | SDPD | HRST | CRST | CRC | PERR | HEAD}

- See Also**
- [":SEARCh:SERial:USBPd:HEADer"](#) on page 1274
 - [":SEARCh:SERial:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERial:USBPd:HEADer:DMESsage"](#) on page 1278
 - [":SEARCh:SERial:USBPd:HEADer:EMESsage"](#) on page 1279
 - [":SEARCh:SERial:USBPd:HEADer:VALue"](#) on page 1282
 - [":SEARCh:SERial:USBPd:HEADer:QUALifier"](#) on page 1281

:SEARCh:SERial:USBPd:HEADer

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:HEADer <type>

<type> ::= {CMESsage | DMESsage | EMESsage | VALue}

When the Header Content search mode is selected (with the ":SEARCh:SERial:USBPd:MODE HEADer" command), the :SEARCh:SERial:USBPd:HEADer command selects the header type:

- CMESsage – Control Message, search on control message types (0 data object).

When the CMESsage header type is selected, use the :SEARCh:SERial:USBPd:HEADer:CMESsage command to select the control message type.

- DMESsage – Data Message, search on data message types (1 or more data objects).

When the DMESsage header type is selected, use the :SEARCh:SERial:USBPd:HEADer:DMESsage command to select the data message type.

- EMESsage – Extended Message, search on extended message types (bit 15 is set).

When the EMESsage header type is selected, use the :SEARCh:SERial:USBPd:HEADer:EMESsage command to select the extended message type.

- VALue – Value, search on a user-defined header value.

When the CMESsage header type is selected, use the :SEARCh:SERial:USBPd:HEADer:VALue command to specify the user-defined header value.

Query Syntax :SEARCh:SERial:USBPd:HEADer?

The :SEARCh:SERial:USBPd:HEADer? query returns the selected Header Content header type

Return Format <type><NL>

<type> ::= {CMES | DMES | EMES | VAL}

- See Also**
- [":SEARCh:SERial:USBPd:MODE"](#) on page 1273
 - [":SEARCh:SERial:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERial:USBPd:HEADer:DMESsage"](#) on page 1278
 - [":SEARCh:SERial:USBPd:HEADer:EMESsage"](#) on page 1279
 - [":SEARCh:SERial:USBPd:HEADer:VALue"](#) on page 1282

- **":SEARch:SERIal:USBPd:HEADer:QUALifier"** on page 1281

:SEARCh:SERial:USBPd:HEADer:CMESsage

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:HEADer:CMESsage <type>

```
<type> ::= {GOODcrc | GOTOmin | ACcept | REJect | PING | PSRDy
            | GSRCap | GSNcap | DRSwap | PRSWap | VCSwap | WAIT | SRST | GSCX
            | GSTatus | FRSWap | GPSTatus | GCCodes}
```

When the Control Message header type is selected (with the ":SEARCh:SERial:USBPd:HEADer:CMESsage" command), the :SEARCh:SERial:USBPd:HEADer:CMESsage command selects the control message type:

- GOODcrc – GoodCRC
- GOTOmin – GotoMin
- ACcept – Accept
- REJect – Reject
- PING – Ping
- PSRDy – PS_RDY
- GSRCap – Get_Source_Cap
- GSNcap – Get_Sink_Cap
- DRSwap – DR_Swap
- PRSWap – PR_Swap
- VCSwap – VCONN_Swap
- WAIT – Wait
- SRST – Soft_Reset
- GSCX – Get_Source_Cap_Extended
- GSTatus – Get_Status
- FRSWap – FR_Swap
- GPSTatus – Get_PPS_Status
- GCCodes – Get_Country_Codes

Query Syntax :SEARCh:SERial:USBPd:HEADer:CMESsage?

The :SEARCh:SERial:USBPd:HEADer:CMESsage? query returns the selected Header Content control message type.

Return Format <type><NL>

```
<type> ::= {GOOD | GOTO | ACC | REJ | PING | PSRD | GSRC | GSNc | DRSW
            | PRSW | VCSW | WAIT | SRST | GSCX | GST | FRSW | GPST | GCC}
```

See Also • [":SEARCh:SERial:USBPd:MODE"](#) on page 1273

- `":SEARCh:SERial:USBPd:HEADer"` on page 1274
- `":SEARCh:SERial:USBPd:HEADer:DMESsage"` on page 1278
- `":SEARCh:SERial:USBPd:HEADer:EMESsage"` on page 1279
- `":SEARCh:SERial:USBPd:HEADer:VALue"` on page 1282
- `":SEARCh:SERial:USBPd:HEADer:QUALifier"` on page 1281

:SEARCh:SERIal:USBPd:HEADer:DMESsage

N (see [page 1708](#))

Command Syntax :SEARCh:SERIal:USBPd:HEADer:DMESsage <type>

<type> ::= {SRCap | REQuest | BIST | SNCap | BStatus | ALERt | GCINfo
| VDEFined}

When the Header Content search mode is selected (with the ":SEARCh:SERIal:USBPd:MODE HEADER" command) and the Data Message header type is selected (with the ":SEARCh:SERIal:USBPd:HEADer DMESsage" command), the :SEARCh:SERIal:USBPd:HEADer:DMESsage command selects the data message type:

- SRCap – Source_Capabilities
- REQuest – Request
- BIST – BIST
- SNCap – Sink_Capabilities
- BStatus – Battery_Status
- ALERt – Alert
- GCINfo – Get_Country_Info
- VDEFined – Vendor_Defined

Query Syntax :SEARCh:SERIal:USBPd:HEADer:DMESsage?

The :SEARCh:SERIal:USBPd:HEADer:DMESsage? query returns the selected Header Content data message type,

Return Format <type><NL>

<type> ::= {SRC | REQ | BIST | SNC | BST | ALER | GCIN | VDEF}

- See Also**
- [":SEARCh:SERIal:USBPd:MODE"](#) on page 1273
 - [":SEARCh:SERIal:USBPd:HEADer"](#) on page 1274
 - [":SEARCh:SERIal:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERIal:USBPd:HEADer:EMESsage"](#) on page 1279
 - [":SEARCh:SERIal:USBPd:HEADer:VALue"](#) on page 1282
 - [":SEARCh:SERIal:USBPd:HEADer:QUALifier"](#) on page 1281

:SEARCh:SERial:USBPd:HEADer:EMESsage

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:HEADer:EMESsage <type>

```
<type> ::= {SCX | STATus | GBCap | GBSTatus | BCAP | GMINfo | MINFo
           | SREQuest | SRESponse | FREQuest | FRESponse | PSTatus | CINfo
           | CCODes}
```

When the Header Content search mode is selected (with the ":SEARCh:SERial:USBPd:MODE HEADer" command) and the Extended Message header type is selected (with the ":SEARCh:SERial:USBPd:HEADer EMESsage" command), the :SEARCh:SERial:USBPd:HEADer:EMESsage command selects the extended message type:

- SCX – Source_Capabilities_Extended
- STATus – Status
- GBCap – Get_Battery_Cap
- GBSTatus – Get_Battery_Status
- BCAP – Battery_Capabilities
- GMINfo – Get_Manufacturer_Info
- MINFo – Manufacturer_Info
- SREQuest – Security_Request
- SRESponse – Security_Response
- FREQuest – Firmware_Update_Request
- FRESponse – Firmware_Update_Response
- PSTatus – PPS_Status
- CINfo – Country_Info
- CCODes – Country_Codes

Query Syntax :SEARCh:SERial:USBPd:HEADer:EMESsage?

The :SEARCh:SERial:USBPd:HEADer:EMESsage? query returns the selected Header Content extended message type.

Return Format <type><NL>

```
<type> ::= {SCX | STAT | GBC | GBST | BCAP | GMIN | MINF | SREQ | SRES
           | FREQ | FRES | PST | CINF | CCOD}
```

- See Also**
- [":SEARCh:SERial:USBPd:MODE"](#) on page 1273
 - [":SEARCh:SERial:USBPd:HEADer"](#) on page 1274
 - [":SEARCh:SERial:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERial:USBPd:HEADer:DMESsage"](#) on page 1278

- **":SEARCh:SERial:USBPd:HEADer:VALue"** on page 1282
- **":SEARCh:SERial:USBPd:HEADer:QUALifier"** on page 1281

:SEARCh:SERial:USBPd:HEADer:QUALifier

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:HEADer:QUALifier <type>

<type> ::= {NONE | SOP | SPRime | SDPRime}

The :SEARCh:SERial:USBPd:HEADer:QUALifier command ...

When the Header Content search mode is selected (with the :SEARCh:SERial:USBPd:MODE HEADer command), the :SEARCh:SERial:USBPd:HEADer:QUALifier command selects an additional qualifier for the Header Content search:

- NONE – None. There is no additional qualifier for the trigger.
- SOP – SOP. The trigger occurs on Sync-1, Sync-1, Sync-1, Sync-2 ordered sets only.
- SPRime – SOP'. The trigger occurs on Sync-1, Sync-1, Sync-3, Sync-3 ordered sets only.
- SDPRime – SOP". The trigger occurs on Sync-1, Sync-3, Sync-1, Sync-3 ordered sets only.

Query Syntax :SEARCh:SERial:USBPd:HEADer:QUALifier?

The :SEARCh:SERial:USBPd:HEADer:QUALifier? query returns the selected Header Content additional qualifier.

Return Format <type><NL>

<type> ::= {NONE | SOP | SPR | SDPR}

- See Also**
- [":SEARCh:SERial:USBPd:MODE"](#) on page 1273
 - [":SEARCh:SERial:USBPd:HEADer"](#) on page 1274
 - [":SEARCh:SERial:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERial:USBPd:HEADer:DMESsage"](#) on page 1278
 - [":SEARCh:SERial:USBPd:HEADer:EMESsage"](#) on page 1279
 - [":SEARCh:SERial:USBPd:HEADer:VALue"](#) on page 1282

:SEARCh:SERial:USBPd:HEADer:VALue

N (see [page 1708](#))

Command Syntax :SEARCh:SERial:USBPd:HEADer:VALue <string>

<string> ::= "nn...n" where n ::= {0 | 1 | X}

<string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F | X | \$}

When the Header Content search mode is selected (with the ":SEARCh:SERial:USBPd:MODE HEADer" command) and the user-defined Value type is selected (with the ""SEARCh:SERial:USBPd:HEADer VALue" command), the :SBUS<n>:USBPd:TRIGger:HEADer:VALue command specifies the user-defined header value.

Query Syntax :SEARCh:SERial:USBPd:HEADer:VALue?

The :SEARCh:SERial:USBPd:HEADer:VALue? query returns the specified user-defined header value.

Return Format <string><NL>

<string> ::= "nn...n" where n ::= {0 | 1 | X}

- See Also**
- [":SEARCh:SERial:USBPd:MODE"](#) on page 1273
 - [":SEARCh:SERial:USBPd:HEADer"](#) on page 1274
 - [":SEARCh:SERial:USBPd:HEADer:CMESsage"](#) on page 1276
 - [":SEARCh:SERial:USBPd:HEADer:DMESsage"](#) on page 1278
 - [":SEARCh:SERial:USBPd:HEADer:EMESsage"](#) on page 1279
 - [":SEARCh:SERial:USBPd:HEADer:QUALifier"](#) on page 1281

37 :STATus Commands

The STATus commands subsystem controls the status registers defined for the HD3-Series oscilloscopes. See:

- **"Introduction to :STATus Commands"** on page 1283
- **"General :STATus Commands"** on page 1286
- **":STATus:OPERation Commands"** on page 1288
- **":STATus:OPERation:ARM Commands"** on page 1303
- **":STATus:OPERation:HARDware Commands"** on page 1316
- **":STATus:OPERation:LOCal Commands"** on page 1329
- **":STATus:OPERation:MTESt Commands"** on page 1342
- **":STATus:OPERation:OVERload Commands"** on page 1355
- **":STATus:OPERation:OVERload:PFAult Commands"** on page 1368
- **":STATus:OPERation:POWEr Commands"** on page 1381
- **":STATus:TRIGger Commands"** on page 1394
- **":STATus:USER Commands"** on page 1407

Introduction to :STATus Commands

The commands in the STATus commands subsystem control the status registers defined for the HD3-Series oscilloscopes.

The SCPI-standard Standard Event Status Register, Status Byte Register, and Service Request Enable Register are also present in the HD3-Series oscilloscopes, but they are controlled with the common (*) commands: *ESR?, *ESE, *STB?, and *SRE.

The status registers defined for the HD3-Series oscilloscopes follow the SCPI status register model where a Condition register continuously monitors status, Transition Filters allow status transitions to be latched into an Event register, and an Enable register allows events to affect a Summary Message that is sent to a parent in the status register hierarchy.

Every status register defined for the HD3-Series oscilloscopes is actually a collection of three registers and two transition filters that let you manage particular status events.

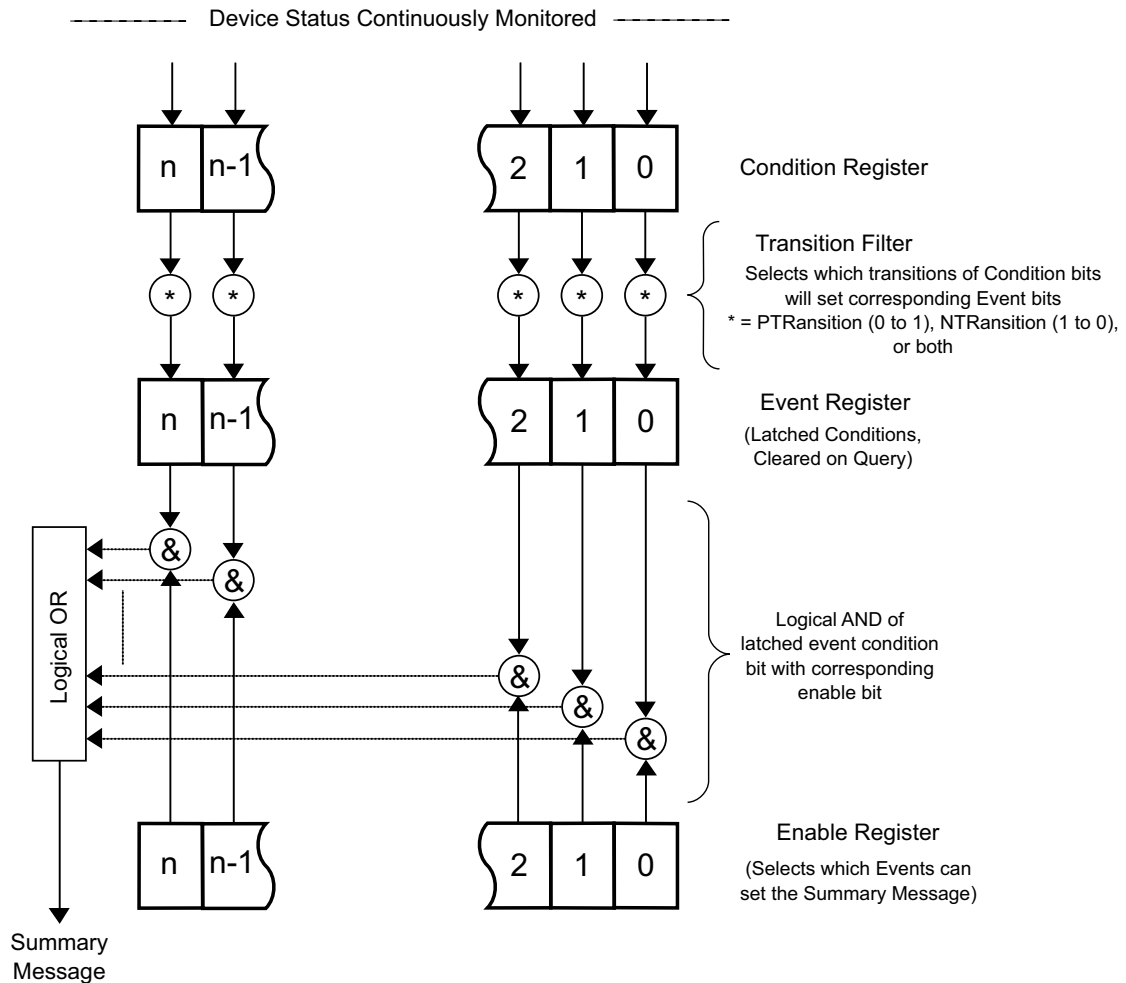


Figure 3 SCPI Status Register Model

So, for each status register defined for the HD3-Series oscilloscopes, there are these commands and queries:

- :STATus:<register>:CONDition? – For reading the Condition register.
- :STATus:<register>:PTRansition – For specifying the bits on which a positive transition (0 to 1 or FALSE to TRUE) will be latched into the Event register.
- :STATus:<register>:NTRansition – For specifying the bits on which a negative transition (1 to 0 or TRUE to FALSE) will be latched into the Event register.

Both positive and negative transitions can be enabled for a bit. In this case, any transition is latched into the Event register.

- :STATus:<register>[:EVENT]? – For reading whether events have occurred since the last time the register was read.

Reading an Event register clears it.

- :STATus:<register>:ENABle — For specifying which bits affect the Summary Message.

And there are these variations for reading and setting individual bits:

- :STATus:<register>:BIT:CONDition?
- :STATus:<register>:BIT:PTRansition
- :STATus:<register>:BIT:NTRansition
- :STATus:<register>:BIT[:EVENT]?

Reading an individual bit in the Event register clears only that bit.

- :STATus:<register>:BIT:ENABle

The status registers defined for the HD3-Series oscilloscopes are 16-bit registers, but the most-significant bit (bit 15) is never used and always returns zero.

- See Also
- **"*ESR? (Standard Event Status Register)"** on page 239
 - **"*ESE (Standard Event Status Enable)"** on page 237
 - **"*STB? (Read Status Byte)"** on page 253
 - **"*SRE (Service Request Enable)"** on page 251

General :STATus Commands

Table 144 General :STATus Commands Summary

Command	Query	Options and Query Returns
:STATus:PRESet (see page 1287)	n/a	n/a

:STATus:PRESet

N (see [page 1708](#))

Command Syntax :STATus:PRESet

The :STATus:PRESet command sets all the transition filters and enable registers to their default state.

- See Also**
- ["Introduction to :STATus Commands"](#) on page 1283
 - ["General :STATus Commands"](#) on page 1286
 - [":STATus:OPERation Commands"](#) on page 1288
 - [":STATus:OPERation:ARM Commands"](#) on page 1303
 - [":STATus:OPERation:HARDware Commands"](#) on page 1316
 - [":STATus:OPERation:LOCal Commands"](#) on page 1329
 - [":STATus:OPERation:MTESt Commands"](#) on page 1342
 - [":STATus:OPERation:OVERload Commands"](#) on page 1355
 - [":STATus:OPERation:OVERload:PFAult Commands"](#) on page 1368
 - [":STATus:OPERation:POWEr Commands"](#) on page 1381
 - [":STATus:TRIGger Commands"](#) on page 1394
 - [":STATus:USER Commands"](#) on page 1407
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation Commands

The :STATus:OPERation commands control the Operation Status Register.

Table 145 :STATus:OPERation Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:BIT :CONDition? (see page 1293)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :ENABle <bit_value> (see page 1294)	:STATus:OPERation:BIT :ENABle? (see page 1294)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :NTRansition <bit_value> (see page 1295)	:STATus:OPERation:BIT :NTRansition? (see page 1295)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:BIT :PTRansition <bit_value> (see page 1296)	:STATus:OPERation:BIT :PTRansition? (see page 1296)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:BIT [:EVENT]? (see page 1297)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:CON Dition? (see page 1298)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ENA Ble <sum_of_bits> (see page 1299)	:STATus:OPERation:ENA Ble? (see page 1299)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:NTR ansition <sum_of_bits> (see page 1300)	:STATus:OPERation:NTR ansition? (see page 1300)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 145 :STATus:OPERation Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:PTR ansition <sum_of_bits> (see page 1301)	:STATus:OPERation:PTR ansition? (see page 1301)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation[:EV ENT]? (see page 1302)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Operation Status Register To see where the Operation Status Register appears in the status register hierarchy, see **Chapter 46**, “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see **"Introduction to :STATus Commands"** on page 1283.

The following bits appear in the Operation Status Register.

Table 146 Operation Status Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:	From:
15	---	---	(Not used, always zero.)	---
14	IOF	IO Operation Failed	Event only when something causes the IO operation to fail, like disconnecting a USB device or interrupting the operation from the oscilloscope's front panel.	---
13	IOC	IO Operation Complete	Event when any IO operation completes. IO operations are any remote data request using any interface (USB or LAN). For example, if you connect to an oscilloscope using the USB interface and then request waveform data, the IOC bit will be set when the IO operation completes.	---
12	HWE	Hardware Event	Event when hardware event occurs.	Hardware Operation Register
11	OVLr	Overload	A 50Ω input overload has occurred.	Overload Operation Register
10	HIST	Histogram	A histogram counter overflow has occurred.	---
9	MTE	Mask Test Event	A mask test event has occurred.	Mask Test Operation Register
8	LCL	Local	A Test Fail or Test Complete has occurred.	Local Operation Register
7	Power	Power	Can potentially indicate a power measurements application Deskew, Apply, or Setup is complete,	Power Operation Register
6	---	---	(Not used.)	---
5	Wait Trig	Wait Trig	The trigger is armed (set by the Trigger Armed Event Register (TER)).	Arm Event Register (can also use :AER?)

Table 146 Operation Status Register Bits (continued)

Bit	Name	Description	When Set (1 = High = True), Indicates:	From:
4	RUI Enab	Remote Enabled	The remote user interface has gone from a disabled state to an enabled state. The front-panel graphical user interface can disable most of the remote interface, including the *OPC syntax, for example when: ▪ a modal dialog box is open ▪ the user is being prompted ▪ all segments are being analyzed ▪ there is a channel overload ▪ certain compliance applications are running When disabled, commands or queries are accepted, but "settings conflict" errors are returned. The status model is the only part of the remote user interface that is enabled. To determine when the remote interface is re-enabled, you can read this bit or wait for the event that gets generated when its status goes from disabled to enabled.	---
3	Run	Running	The oscilloscope has gone from a stop state to a single or running state.	---
2	---	---	(Not used.)	---
1	---	---	(Not used.)	---
0	Calibrating	Calibrating	The oscilloscope is calibrating.	---

When enabled, the summary message for the Operation Status Register goes to Status Byte Register bit 7 (OPER). See "***CLS (Clear Status)**" on page 236.

:STATus:OPERation:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:BIT:ENABLE?

The :STATus:OPERation:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABLE"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:BIT:NTRansition?

The :STATus:OPERation:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:BIT:PTRansition?

The :STATus:OPERation:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:CONDition?

The :STATus:OPERation:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:ENABle?

The :STATus:OPERation:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:NTRansition?

The :STATus:OPERation:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABLE"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABLE"](#) on page 1299
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:PTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATUS:OPERation:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATUS:OPERation:PTRansition?

The :STATUS:OPERation:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATUS:OPERation:BIT:ENABLE"](#) on page 1294
 - [":STATUS:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATUS:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATUS:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATUS:OPERation:CONDition?"](#) on page 1298
 - [":STATUS:OPERation:ENABLE"](#) on page 1299
 - [":STATUS:OPERation:NTRansition"](#) on page 1300
 - [":STATUS:OPERation\[:EVENT\]?"](#) on page 1302
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation[:EVENT]?

The :STATus:OPERation[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:BIT:CONDition?"](#) on page 1293
 - [":STATus:OPERation:BIT:ENABle"](#) on page 1294
 - [":STATus:OPERation:BIT:NTRansition"](#) on page 1295
 - [":STATus:OPERation:BIT:PTRansition"](#) on page 1296
 - [":STATus:OPERation:BIT\[:EVENT\]?"](#) on page 1297
 - [":STATus:OPERation:CONDition?"](#) on page 1298
 - [":STATus:OPERation:ENABle"](#) on page 1299
 - [":STATus:OPERation:NTRansition"](#) on page 1300
 - [":STATus:OPERation:PTRansition"](#) on page 1301
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM Commands

The :STATus:OPERation:ARM commands control the Arm Event Register.

Table 147 :STATus:OPERation:ARM Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:ARM :BIT:CONDition? (see page 1306)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:ENABle <bit_value> (see page 1307)	:STATus:OPERation:ARM :BIT:ENABle? (see page 1307)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:NTRansition <bit_value> (see page 1308)	:STATus:OPERation:ARM :BIT:NTRansition? (see page 1308)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :BIT:PTRansition <bit_value> (see page 1309)	:STATus:OPERation:ARM :BIT:PTRansition? (see page 1309)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM :BIT[:EVENT]? (see page 1310)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM :CONDition? (see page 1311)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :ENABle <sum_of_bits> (see page 1312)	:STATus:OPERation:ARM :ENABle? (see page 1312)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:ARM :NTRansition <sum_of_bits> (see page 1313)	:STATus:OPERation:ARM :NTRansition? (see page 1313)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 147 :STATus:OPERation:ARM Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:ARM :PTRansition <sum_of_bits> (see page 1314)	:STATus:OPERation:ARM :PTRansition? (see page 1314)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:ARM [:EVENT]? (see page 1315)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Arm Event Register To see where the Arm Event Register appears in the status register hierarchy, see **Chapter 46**, “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see **"Introduction to :STATus Commands"** on page 1283.

The following bits appear in the Arm Event Register.

Table 148 Arm Event Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	---	---	(Not used.)
1	---	---	(Not used.)
0	Armed	Armed	The trigger system is ready to look of triggers.

When enabled, the summary message for the Arm Event Register goes to Operation Status Register bit 5 (WAIT TRIG). See **":STATus:OPERation Commands"** on page 1288.

:STATus:OPERation:ARM:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:ARM:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:ARM:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABle"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ARM:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:ARM:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:ARM:BIT:ENABLE?

The :STATus:OPERation:ARM:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABLE"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ARM:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:ARM:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:ARM:BIT:NTRansition?

The :STATus:OPERation:ARM:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABle"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ARM:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:ARM:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:ARM:BIT:PTRansition?

The :STATus:OPERation:ARM:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABle"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:ARM:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:ARM:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

NOTE

The :AER? query is an alias for the :STATus:OPERation:ARM:BIT0[:EVENT]? query.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABle"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":AER? \(Arm Event Register\)"](#) on page 263

:STATus:OPERation:ARM:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:ARM:CONDition?

The :STATus:OPERation:ARM:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABLE"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:ENABLE"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ARM:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:ARM:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:ARM:ENABle?

The :STATus:OPERation:ARM:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:ARM:NTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:ARM:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATUS:OPERation:ARM:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATUS:OPERation:ARM:NTRansition?

The :STATUS:OPERation:ARM:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATUS:OPERation:ARM:BIT:ENABLE"](#) on page 1307
 - [":STATUS:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATUS:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATUS:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATUS:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATUS:OPERation:ARM:ENABLE"](#) on page 1312
 - [":STATUS:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATUS:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:ARM:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:ARM:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:OPERation:ARM:PTRansition?

The :STATus:OPERation:ARM:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABLE"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABLE"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM\[:EVENT\]?"](#) on page 1315
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:ARM[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:ARM[:EVENT] ?

The :STATus:OPERation:ARM[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

NOTE

Because bit 0 is the only bit being used in the Arm Event Register and the only one that can be latched into the Event register, the :AER? query is an alias for the :STATus:OPERation:ARM[:EVENT]? query.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:ARM:BIT:CONDition?"](#) on page 1306
 - [":STATus:OPERation:ARM:BIT:ENABle"](#) on page 1307
 - [":STATus:OPERation:ARM:BIT:NTRansition"](#) on page 1308
 - [":STATus:OPERation:ARM:BIT:PTRansition"](#) on page 1309
 - [":STATus:OPERation:ARM:BIT\[:EVENT\]?"](#) on page 1310
 - [":STATus:OPERation:ARM:CONDition?"](#) on page 1311
 - [":STATus:OPERation:ARM:ENABle"](#) on page 1312
 - [":STATus:OPERation:ARM:NTRansition"](#) on page 1313
 - [":STATus:OPERation:ARM:PTRansition"](#) on page 1314
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":AER? \(Arm Event Register\)"](#) on page 263

:STATus:OPERation:HARDware Commands

The :STATus:OPERation:HARDware commands control the Hardware Operation Register.

Table 149 :STATus:OPERation:HARDware Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:HARDware:BIT:CONDition? (see page 1319)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:BIT:ENABLE <bit_value> (see page 1320)	:STATus:OPERation:HARDware:BIT:ENABLE? (see page 1320)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:BIT:NTRansition <bit_value> (see page 1321)	:STATus:OPERation:HARDware:BIT:NTRansition? (see page 1321)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:BIT:PTRansition <bit_value> (see page 1322)	:STATus:OPERation:HARDware:BIT:PTRansition? (see page 1322)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HARDware:BIT[:EVENT]? (see page 1323)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HARDware:CONDition? (see page 1324)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:ENABLE <sum_of_bits> (see page 1325)	:STATus:OPERation:HARDware:ENABLE? (see page 1325)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:HARDware:NTRansition <sum_of_bits> (see page 1326)	:STATus:OPERation:HARDware:NTRansition? (see page 1326)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 149 :STATus:OPERation:HARDware Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:HARDware:PTRansition <sum_of_bits> (see page 1327)	:STATus:OPERation:HARDware:PTRansition? (see page 1327)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:HARDware[:EVENT]? (see page 1328)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Hardware Operation Register To see where the Hardware Operation Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Hardware Operation Register.

Table 150 Hardware Operation Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	PLL Locked	PLL Locked	This bit is for internal use and is not intended for general use.
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	---	---	(Not used.)
1	Battery Low	Battery Low	Reserved for battery-operated oscilloscope models.
0	Battery On	Battery On	Reserved for battery-operated oscilloscope models.

When enabled, the summary message for the Hardware Operation Register goes to Operation Status Register bit 12 (HWE). See **":STATus:OPERation Commands"** on page 1288.

:STATus:OPERation:HARDware:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:HARDware:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:HARDware:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:HARDware:BIT:ENABLE"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABLE"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:HARDware:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:HARDware:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:HARDware:BIT:ENABLE?

The :STATus:OPERation:HARDware:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABLE"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:HARDware:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:HARDware:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATUS:OPERation:HARDware:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATUS:OPERation:HARDware:BIT:NTRansition?

The :STATUS:OPERation:HARDware:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATUS:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATUS:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATUS:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATUS:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATUS:OPERation:HARDware:ENABle"](#) on page 1325
 - [":STATUS:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATUS:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATUS:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:HARDware:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:HARDware:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:HARDware:BIT:PTRansition?

The :STATus:OPERation:HARDware:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABle"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:HARDware:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:HARDware:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABle"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:HARDware:CONDition?

The :STATus:OPERation:HARDware:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:ENABle"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:HARDware:ENABle <sum_of_bits>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:HARDware:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:HARDware:ENABle?

The :STATus:OPERation:HARDware:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:HARDware:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:HARDware:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:HARDware:NTRansition?

The :STATus:OPERation:HARDware:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABLE"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABLE"](#) on page 1325
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:HARDware:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:HARDware:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:OPERation:HARDware:PTRansition?

The :STATus:OPERation:HARDware:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABLE"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABLE"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware\[:EVENT\]?"](#) on page 1328
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:HARDware[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:HARDware[:EVENT]?

The :STATus:OPERation:HARDware[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:HARDware:BIT:CONDition?"](#) on page 1319
 - [":STATus:OPERation:HARDware:BIT:ENABle"](#) on page 1320
 - [":STATus:OPERation:HARDware:BIT:NTRansition"](#) on page 1321
 - [":STATus:OPERation:HARDware:BIT:PTRansition"](#) on page 1322
 - [":STATus:OPERation:HARDware:BIT\[:EVENT\]?"](#) on page 1323
 - [":STATus:OPERation:HARDware:CONDition?"](#) on page 1324
 - [":STATus:OPERation:HARDware:ENABle"](#) on page 1325
 - [":STATus:OPERation:HARDware:NTRansition"](#) on page 1326
 - [":STATus:OPERation:HARDware:PTRansition"](#) on page 1327
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal Commands

The :STATus:OPERation:LOCal commands control the Local Operation Register.

Table 151 :STATus:OPERation:LOCAl Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:LOCAl:BIT:CONDition? (see page 1332)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:ENABle <bit_value> (see page 1333)	:STATus:OPERation:LOCAl:BIT:ENABle? (see page 1333)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:NTRansition <bit_value> (see page 1334)	:STATus:OPERation:LOCAl:BIT:NTRansition? (see page 1334)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:BIT:PTRansition <bit_value> (see page 1335)	:STATus:OPERation:LOCAl:BIT:PTRansition? (see page 1335)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:LOCAl:BIT[:EVENT]? (see page 1336)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:LOCAl:CONDition? (see page 1337)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:ENABle <sum_of_bits> (see page 1338)	:STATus:OPERation:LOCAl:ENABle? (see page 1338)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:LOCAl:NTRansition <sum_of_bits> (see page 1339)	:STATus:OPERation:LOCAl:NTRansition? (see page 1339)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 151 :STATus:OPERation:LOCal Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:LOC al:PTRansition <sum_of_bits> (see page 1340)	:STATus:OPERation:LOC al:PTRansition? (see page 1340)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:LOC al[:EVENTt]? (see page 1341)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Local Operation Register To see where the Local Operation Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Local Operation Register.

Table 152 Local Operation Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	---	---	(Not used.)
1	Test Fail	Test Fail	A test failure has occurred.
0	Test Complete	Test Complete	A test completion has occurred.

When enabled, the summary message for the Local Operation Register goes to Operation Status Register bit 8 (Lcl). See **":STATus:OPERation Commands"** on page 1288.

:STATus:OPERation:LOCal:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:LOCal:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:LOCal:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:LOCal:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:LOCal:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:LOCal:BIT:ENABLE?

The :STATus:OPERation:LOCal:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABLE"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:LOCal:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:LOCal:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:LOCal:BIT:NTRansition?

The :STATus:OPERation:LOCal:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:LOCal:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:LOCal:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:LOCal:BIT:PTRansition?

The :STATus:OPERation:LOCal:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:LOCal:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:LOCal:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:LOCAl:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCAl:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCAl:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCAl:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCAl:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCAl:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCAl:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCAl:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCAl\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:LOCal:CONDition?

The :STATus:OPERation:LOCal:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:LOCAl:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCAl:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCAl:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCAl:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCAl:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCAl:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCAl:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCAl:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCAl\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:LOCal:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:LOCal:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:LOCal:ENABle?

The :STATus:OPERation:LOCal:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:LOCAL:NTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:LOCAL:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATUS:OPERation:LOCAL:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATUS:OPERation:LOCAL:NTRansition?

The :STATUS:OPERation:LOCAL:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:LOCAL:BIT:CONDition?"](#) on page 1332
 - [":STATUS:OPERation:LOCAL:BIT:ENABLE"](#) on page 1333
 - [":STATUS:OPERation:LOCAL:BIT:NTRansition"](#) on page 1334
 - [":STATUS:OPERation:LOCAL:BIT:PTRansition"](#) on page 1335
 - [":STATUS:OPERation:LOCAL:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATUS:OPERation:LOCAL:CONDition?"](#) on page 1337
 - [":STATUS:OPERation:LOCAL:ENABLE"](#) on page 1338
 - [":STATUS:OPERation:LOCAL:PTRansition"](#) on page 1340
 - [":STATUS:OPERation:LOCAL\[:EVENT\]?"](#) on page 1341
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:LOCal:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:LOCal:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:OPERation:LOCal:PTRansition?

The :STATus:OPERation:LOCal:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal\[:EVENT\]?"](#) on page 1341
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:LOCal[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:LOCal[:EVENT]?

The :STATus:OPERation:LOCal[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:LOCal:BIT:CONDition?"](#) on page 1332
 - [":STATus:OPERation:LOCal:BIT:ENABle"](#) on page 1333
 - [":STATus:OPERation:LOCal:BIT:NTRansition"](#) on page 1334
 - [":STATus:OPERation:LOCal:BIT:PTRansition"](#) on page 1335
 - [":STATus:OPERation:LOCal:BIT\[:EVENT\]?"](#) on page 1336
 - [":STATus:OPERation:LOCal:CONDition?"](#) on page 1337
 - [":STATus:OPERation:LOCal:ENABle"](#) on page 1338
 - [":STATus:OPERation:LOCal:NTRansition"](#) on page 1339
 - [":STATus:OPERation:LOCal:PTRansition"](#) on page 1340
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTESt Commands

The :STATus:OPERation:MTESt commands control the Mask Test Operation Register.

Table 153 :STATus:OPERation:MTESt Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:MTESt:BIT:CONDition? (see page 1345)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:BIT:ENABle <bit_value> (see page 1346)	:STATus:OPERation:MTESt:BIT:ENABle? (see page 1346)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:BIT:NTRansition <bit_value> (see page 1347)	:STATus:OPERation:MTESt:BIT:NTRansition? (see page 1347)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:BIT:PTRansition <bit_value> (see page 1348)	:STATus:OPERation:MTESt:BIT:PTRansition? (see page 1348)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:MTESt:BIT[:EVENT]? (see page 1349)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:MTESt:CONDition? (see page 1350)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:ENABle <sum_of_bits> (see page 1351)	:STATus:OPERation:MTESt:ENABle? (see page 1351)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:MTESt:NTRansition <sum_of_bits> (see page 1352)	:STATus:OPERation:MTESt:NTRansition? (see page 1352)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 153 :STATus:OPERation:MTEst Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:MTESt:PTRansition <sum_of_bits> (see page 1353)	:STATus:OPERation:MTESt:PTRansition? (see page 1353)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:MTESt[:EVENTt]? (see page 1354)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Mask Test Operation Register To see where the Mask Test Operation Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Mask Test Operation Register.

Table 154 Mask Test Operation Register Bits

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

When enabled, the summary message for the Mask Test Operation Register goes to Operation Status Register bit 9 (MTE). See **"[:STATus:OPERation Commands](#)"** on page 1288.

:STATus:OPERation:MTESt:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:MTESt:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:MTESt:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:MTESt:BIT:ENABle"](#) on page 1346
 - [":STATus:OPERation:MTESt:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTESt:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTESt:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTESt:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTESt:ENABle"](#) on page 1351
 - [":STATus:OPERation:MTESt:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTESt:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTESt\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:MTEST:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:MTEST:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:MTEST:BIT:ENABLE?

The :STATus:OPERation:MTEST:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:MTEST:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:MTEST:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:MTEST:BIT:NTRansition?

The :STATus:OPERation:MTEST:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABLE"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:MTEST:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:MTEST:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:MTEST:BIT:PTRansition?

The :STATus:OPERation:MTEST:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABLE"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:MTEST:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:MTEST:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABLE"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:MTEST:CONDition?

The :STATus:OPERation:MTEST:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABLE"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:MTEST:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:MTEST:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:MTEST:ENABle?

The :STATus:OPERation:MTEST:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABle"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTESt:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:MTESt:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:MTESt:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:MTESt:NTRansition?

The :STATus:OPERation:MTESt:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:MTESt:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTESt:BIT:ENABLE"](#) on page 1346
 - [":STATus:OPERation:MTESt:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTESt:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTESt:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTESt:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTESt:ENABLE"](#) on page 1351
 - [":STATus:OPERation:MTESt:PTRansition"](#) on page 1353
 - [":STATus:OPERation:MTESt\[:EVENT\]?"](#) on page 1354
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:MTEST:PTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:MTEST:PTRansition <sum_of_bits>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format
 ::= 0-14; an integer in NR1 format

The :STATUS:OPERation:MTEST:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATUS:OPERation:MTEST:PTRansition?

The :STATUS:OPERation:MTEST:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATUS:OPERation:MTEST:BIT:ENABLE"](#) on page 1346
 - [":STATUS:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATUS:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATUS:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATUS:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATUS:OPERation:MTEST:ENABLE"](#) on page 1351
 - [":STATUS:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATUS:OPERation:MTEST\[:EVENT\]?"](#) on page 1354
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:MTEST[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:MTEST[:EVENT]?

The :STATus:OPERation:MTEST[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:MTEST:BIT:CONDition?"](#) on page 1345
 - [":STATus:OPERation:MTEST:BIT:ENABle"](#) on page 1346
 - [":STATus:OPERation:MTEST:BIT:NTRansition"](#) on page 1347
 - [":STATus:OPERation:MTEST:BIT:PTRansition"](#) on page 1348
 - [":STATus:OPERation:MTEST:BIT\[:EVENT\]?"](#) on page 1349
 - [":STATus:OPERation:MTEST:CONDition?"](#) on page 1350
 - [":STATus:OPERation:MTEST:ENABle"](#) on page 1351
 - [":STATus:OPERation:MTEST:NTRansition"](#) on page 1352
 - [":STATus:OPERation:MTEST:PTRansition"](#) on page 1353
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload Commands

The :STATus:OPERation:OVERload commands control the Overload Operation Register.

Table 155 :STATus:OPERation:OVERload Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:OVERload:BIT:CONDition? (see page 1358)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:BIT:ENABLE <bit_value> (see page 1359)	:STATus:OPERation:OVERload:BIT:ENABLE? (see page 1359)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:BIT:NTRansition <bit_value> (see page 1360)	:STATus:OPERation:OVERload:BIT:NTRansition? (see page 1360)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:BIT:PTRansition <bit_value> (see page 1361)	:STATus:OPERation:OVERload:BIT:PTRansition? (see page 1361)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:BIT[:EVENT]? (see page 1362)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:CONDition? (see page 1363)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:ENABLE <sum_of_bits> (see page 1364)	:STATus:OPERation:OVERload:ENABLE? (see page 1364)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:NTRansition <sum_of_bits> (see page 1365)	:STATus:OPERation:OVERload:NTRansition? (see page 1365)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 155 :STATus:OPERation:OVERload Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:OVERload:PTRansition <sum_of_bits> (see page 1366)	:STATus:OPERation:OVERload:PTRansition? (see page 1366)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload[:EVENT]? (see page 1367)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Overload Operation Register To see where the Overload Operation Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Overload Operation Register.

Table 156 Overload Operation Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:	From:
15	---	---	(Not used, always zero.)	---
14	---	---	(Not used.)	---
13	---	External Trigger Fault	Fault has occurred on the External Trigger input.	---
12	---	Channel Probe Fault Summary	Enabled event in the Probe Fault Register has occurred.	Probe Fault Register
11	---	---	(Not used.)	---
10	---	---	(Not used.)	---
9	---	---	(Not used.)	---
8	EXT TRIG OVLD	External Trigger Overload	Overload has occurred on the External Trigger input.	---
7	CH8 OVLD	Channel 8 Overload.	Overload has occurred on Channel 8 input.	---
6	CH7 OVLD	Channel 7 Overload.	Overload has occurred on Channel 7 input.	---
5	CH6 OVLD	Channel 6 Overload.	Overload has occurred on Channel 6 input.	---
4	CH5 OVLD	Channel 5 Overload.	Overload has occurred on Channel 5 input.	---
3	CH4 OVLD	Channel 4 Overload.	Overload has occurred on Channel 4 input.	---
2	CH3 OVLD	Channel 3 Overload.	Overload has occurred on Channel 3 input.	---
1	CH2 OVLD	Channel 2 Overload.	Overload has occurred on Channel 2 input.	---
0	CH1 OVLD	Channel 1 Overload.	Overload has occurred on Channel 1 input.	---

When enabled, the summary message for the Overload Operation Register goes to Operation Status Register bit 11 (OVLD). See **":STATus:OPERation Commands"** on page 1288.

:STATus:OPERation:OVERload:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:OVERload:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:BIT:ENABle"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABle"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:OVERload:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:OVERload:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATUS:OPERation:OVERload:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATUS:OPERation:OVERload:BIT:ENABLE?

The :STATUS:OPERation:OVERload:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATUS:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATUS:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATUS:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATUS:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATUS:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATUS:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATUS:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATUS:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:OVERload:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:OVERload:BIT:NTRansition?

The :STATus:OPERation:OVERload:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:OVERload:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:OVERload:BIT:PTRansition?

The :STATus:OPERation:OVERload:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABle"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABle"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:OVERload:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABle"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABle"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload:CONDition?

The :STATus:OPERation:OVERload:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:ENABLE <sum_of_bits>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:OVERload:ENABLE command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:OVERload:ENABLE?

The :STATus:OPERation:OVERload:ENABLE? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>
 <sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:OVERload:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:OVERload:NTRansition?

The :STATus:OPERation:OVERload:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:OVERload:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:OPERation:OVERload:PTRansition?

The :STATus:OPERation:OVERload:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload\[:EVENT\]?"](#) on page 1367
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload[:EVENT] ?

The :STATus:OPERation:OVERload[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:BIT:CONDition?"](#) on page 1358
 - [":STATus:OPERation:OVERload:BIT:ENABLE"](#) on page 1359
 - [":STATus:OPERation:OVERload:BIT:NTRansition"](#) on page 1360
 - [":STATus:OPERation:OVERload:BIT:PTRansition"](#) on page 1361
 - [":STATus:OPERation:OVERload:BIT\[:EVENT\]?"](#) on page 1362
 - [":STATus:OPERation:OVERload:CONDition?"](#) on page 1363
 - [":STATus:OPERation:OVERload:ENABLE"](#) on page 1364
 - [":STATus:OPERation:OVERload:NTRansition"](#) on page 1365
 - [":STATus:OPERation:OVERload:PTRansition"](#) on page 1366
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFAult Commands

The :STATus:OPERation:OVERload:PFAult commands control the Probe Fault Register.

Table 157 :STATus:OPERation:OVERload:PFAult Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:OVERload:PFAult:BIT:CONDITION? (see page 1371)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:BIT:ENABLE <bit_value> (see page 1372)	:STATus:OPERation:OVERload:PFAult:BIT:ENABLE? (see page 1372)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:BIT:NTransition <bit_value> (see page 1373)	:STATus:OPERation:OVERload:PFAult:BIT:NTransition? (see page 1373)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:BIT:PTransition <bit_value> (see page 1374)	:STATus:OPERation:OVERload:PFAult:BIT:PTransition? (see page 1374)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFAult:BIT[:EVENT]? (see page 1375)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFAult:CONDITION? (see page 1376)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:ENABLE <sum_of_bits> (see page 1377)	:STATus:OPERation:OVERload:PFAult:ENABLE? (see page 1377)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:OVERload:PFAult:NTransition <sum_of_bits> (see page 1378)	:STATus:OPERation:OVERload:PFAult:NTransition? (see page 1378)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 157 :STATus:OPERation:OVERload:PFault Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:OVERload:PFault:PTRansition <sum_of_bits> (see page 1379)	:STATus:OPERation:OVERload:PFault:PTRansition? (see page 1379)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:OVERload:PFault[:EVENT]? (see page 1380)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Probe Fault Register To see where the Probe Fault Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Overload Operation Register.

Table 158 Probe Fault Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	CH8 PFA	Channel 8 Probe Fault.	Fault has occurred on Channel 8 input.
6	CH7 PFA	Channel 7 Probe Fault.	Fault has occurred on Channel 7 input.
5	CH6 PFA	Channel 6 Probe Fault.	Fault has occurred on Channel 6 input.
4	CH5 PFA	Channel 5 Probe Fault.	Fault has occurred on Channel 5 input.
3	CH4 PFA	Channel 4 Probe Fault.	Fault has occurred on Channel 4 input.
2	CH3 PFA	Channel 3 Probe Fault.	Fault has occurred on Channel 3 input.
1	CH2 PFA	Channel 2 Probe Fault.	Fault has occurred on Channel 2 input.
0	CH1 PFA	Channel 1 Probe Fault.	Fault has occurred on Channel 1 input.

When enabled, the summary message for the Probe Fault Register goes to Overload Operation Register bit 12 (Chanel Probe Fault Summary). See **":STATus:OPERation:OVERload Commands"** on page 1355.

:STATUS:OPERation:OVERload:PFault:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATUS:OPERation:OVERload:PFault:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATUS:OPERation:OVERload:PFault:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:OPERation:OVERload:PFault:BIT:ENABLE"](#) on page 1372
 - [":STATUS:OPERation:OVERload:PFault:BIT:NTRansition"](#) on page 1373
 - [":STATUS:OPERation:OVERload:PFault:BIT:PTRansition"](#) on page 1374
 - [":STATUS:OPERation:OVERload:PFault:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATUS:OPERation:OVERload:PFault:CONDition?"](#) on page 1376
 - [":STATUS:OPERation:OVERload:PFault:ENABLE"](#) on page 1377
 - [":STATUS:OPERation:OVERload:PFault:NTRansition"](#) on page 1378
 - [":STATUS:OPERation:OVERload:PFault:PTRansition"](#) on page 1379
 - [":STATUS:OPERation:OVERload:PFault\[:EVENT\]?"](#) on page 1380
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFAult:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:PFAult:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:OVERload:PFAult:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:OVERload:PFAult:BIT:ENABLE?

The :STATus:OPERation:OVERload:PFAult:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:PFAult:BIT:CONDition?"](#) on page 1371
 - [":STATus:OPERation:OVERload:PFAult:BIT:NTRansition"](#) on page 1373
 - [":STATus:OPERation:OVERload:PFAult:BIT:PTRansition"](#) on page 1374
 - [":STATus:OPERation:OVERload:PFAult:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATus:OPERation:OVERload:PFAult:CONDition?"](#) on page 1376
 - [":STATus:OPERation:OVERload:PFAult:ENABLE"](#) on page 1377
 - [":STATus:OPERation:OVERload:PFAult:NTRansition"](#) on page 1378
 - [":STATus:OPERation:OVERload:PFAult:PTRansition"](#) on page 1379
 - [":STATus:OPERation:OVERload:PFAult\[:EVENT\]?"](#) on page 1380
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:OVERload:PFAult:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:OVERload:PFAult:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATUS:OPERation:OVERload:PFAult:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATUS:OPERation:OVERload:PFAult:BIT:NTRansition?

The :STATUS:OPERation:OVERload:PFAult:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:OPERation:OVERload:PFAult:BIT:CONDition?"](#) on page 1371
 - [":STATUS:OPERation:OVERload:PFAult:BIT:ENABle"](#) on page 1372
 - [":STATUS:OPERation:OVERload:PFAult:BIT:PTRansition"](#) on page 1374
 - [":STATUS:OPERation:OVERload:PFAult:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATUS:OPERation:OVERload:PFAult:CONDition?"](#) on page 1376
 - [":STATUS:OPERation:OVERload:PFAult:ENABle"](#) on page 1377
 - [":STATUS:OPERation:OVERload:PFAult:NTRansition"](#) on page 1378
 - [":STATUS:OPERation:OVERload:PFAult:PTRansition"](#) on page 1379
 - [":STATUS:OPERation:OVERload:PFAult\[:EVENT\]?"](#) on page 1380
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFAult:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:PFAult:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:OVERload:PFAult:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:OVERload:PFAult:BIT:PTRansition?

The :STATus:OPERation:OVERload:PFAult:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:OVERload:PFAult:BIT:CONDition?"](#) on page 1371
 - [":STATus:OPERation:OVERload:PFAult:BIT:ENABle"](#) on page 1372
 - [":STATus:OPERation:OVERload:PFAult:BIT:NTRansition"](#) on page 1373
 - [":STATus:OPERation:OVERload:PFAult:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATus:OPERation:OVERload:PFAult:CONDition?"](#) on page 1376
 - [":STATus:OPERation:OVERload:PFAult:ENABle"](#) on page 1377
 - [":STATus:OPERation:OVERload:PFAult:NTRansition"](#) on page 1378
 - [":STATus:OPERation:OVERload:PFAult:PTRansition"](#) on page 1379
 - [":STATus:OPERation:OVERload:PFAult\[:EVENT\]?"](#) on page 1380
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:OVERload:PFault:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATUS:OPERation:OVERload:PFault:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATUS:OPERation:OVERload:PFault:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:OPERation:OVERload:PFault:BIT:CONDition?"](#) on page 1371
 - [":STATUS:OPERation:OVERload:PFault:BIT:ENABLE"](#) on page 1372
 - [":STATUS:OPERation:OVERload:PFault:BIT:NTRansition"](#) on page 1373
 - [":STATUS:OPERation:OVERload:PFault:BIT:PTRansition"](#) on page 1374
 - [":STATUS:OPERation:OVERload:PFault:CONDition?"](#) on page 1376
 - [":STATUS:OPERation:OVERload:PFault:ENABLE"](#) on page 1377
 - [":STATUS:OPERation:OVERload:PFault:NTRansition"](#) on page 1378
 - [":STATUS:OPERation:OVERload:PFault:PTRansition"](#) on page 1379
 - [":STATUS:OPERation:OVERload:PFault\[:EVENT\]?"](#) on page 1380
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFAult:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload:PFAult:CONDition?

The :STATus:OPERation:OVERload:PFAult:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:PFAult:BIT:CONDition?"](#) on page 1371
 - [":STATus:OPERation:OVERload:PFAult:BIT:ENABLE"](#) on page 1372
 - [":STATus:OPERation:OVERload:PFAult:BIT:NTRansition"](#) on page 1373
 - [":STATus:OPERation:OVERload:PFAult:BIT:PTRansition"](#) on page 1374
 - [":STATus:OPERation:OVERload:PFAult:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATus:OPERation:OVERload:PFAult:ENABLE"](#) on page 1377
 - [":STATus:OPERation:OVERload:PFAult:NTRansition"](#) on page 1378
 - [":STATus:OPERation:OVERload:PFAult:PTRansition"](#) on page 1379
 - [":STATus:OPERation:OVERload:PFAult\[:EVENT\]?"](#) on page 1380
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:OVERload:PFault:ENABLE

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:OVERload:PFault:ENABLE <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATUS:OPERation:OVERload:PFault:ENABLE command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATUS:OPERation:OVERload:PFault:ENABLE?

The :STATUS:OPERation:OVERload:PFault:ENABLE? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:OVERload:PFault:BIT:CONDition?"](#) on page 1371
 - [":STATUS:OPERation:OVERload:PFault:BIT:ENABLE"](#) on page 1372
 - [":STATUS:OPERation:OVERload:PFault:BIT:NTRansition"](#) on page 1373
 - [":STATUS:OPERation:OVERload:PFault:BIT:PTRansition"](#) on page 1374
 - [":STATUS:OPERation:OVERload:PFault:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATUS:OPERation:OVERload:PFault:CONDition?"](#) on page 1376
 - [":STATUS:OPERation:OVERload:PFault:NTRansition"](#) on page 1378
 - [":STATUS:OPERation:OVERload:PFault:PTRansition"](#) on page 1379
 - [":STATUS:OPERation:OVERload:PFault\[:EVENT\]?"](#) on page 1380
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFault:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:OVERload:PFault:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:OVERload:PFault:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:OVERload:PFault:NTRansition?

The :STATus:OPERation:OVERload:PFault:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:PFault:BIT:CONDition?"](#) on page 1371
 - [":STATus:OPERation:OVERload:PFault:BIT:ENABLE"](#) on page 1372
 - [":STATus:OPERation:OVERload:PFault:BIT:NTRansition"](#) on page 1373
 - [":STATus:OPERation:OVERload:PFault:BIT:PTRansition"](#) on page 1374
 - [":STATus:OPERation:OVERload:PFault:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATus:OPERation:OVERload:PFault:CONDition?"](#) on page 1376
 - [":STATus:OPERation:OVERload:PFault:ENABLE"](#) on page 1377
 - [":STATus:OPERation:OVERload:PFault:PTRansition"](#) on page 1379
 - [":STATus:OPERation:OVERload:PFault\[:EVENT\]?"](#) on page 1380
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:OPERation:OVERload:PFAult:PTRansition

N (see [page 1708](#))

Command Syntax :STATUS:OPERation:OVERload:PFAult:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATUS:OPERation:OVERload:PFAult:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATUS:OPERation:OVERload:PFAult:PTRansition?

The :STATUS:OPERation:OVERload:PFAult:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATUS:OPERation:OVERload:PFAult:BIT:CONDition?"](#) on page 1371
 - [":STATUS:OPERation:OVERload:PFAult:BIT:ENABLE"](#) on page 1372
 - [":STATUS:OPERation:OVERload:PFAult:BIT:NTRansition"](#) on page 1373
 - [":STATUS:OPERation:OVERload:PFAult:BIT:PTRansition"](#) on page 1374
 - [":STATUS:OPERation:OVERload:PFAult:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATUS:OPERation:OVERload:PFAult:CONDition?"](#) on page 1376
 - [":STATUS:OPERation:OVERload:PFAult:ENABLE"](#) on page 1377
 - [":STATUS:OPERation:OVERload:PFAult:NTRansition"](#) on page 1378
 - [":STATUS:OPERation:OVERload:PFAult\[:EVENT\]?"](#) on page 1380
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:OVERload:PFault[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:OVERload:PFault[:EVENT]?

The :STATus:OPERation:OVERload:PFault[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:OVERload:PFault:BIT:CONDition?"](#) on page 1371
 - [":STATus:OPERation:OVERload:PFault:BIT:ENABLE"](#) on page 1372
 - [":STATus:OPERation:OVERload:PFault:BIT:NTRansition"](#) on page 1373
 - [":STATus:OPERation:OVERload:PFault:BIT:PTRansition"](#) on page 1374
 - [":STATus:OPERation:OVERload:PFault:BIT\[:EVENT\]?"](#) on page 1375
 - [":STATus:OPERation:OVERload:PFault:CONDition?"](#) on page 1376
 - [":STATus:OPERation:OVERload:PFault:ENABLE"](#) on page 1377
 - [":STATus:OPERation:OVERload:PFault:NTRansition"](#) on page 1378
 - [":STATus:OPERation:OVERload:PFault:PTRansition"](#) on page 1379
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer Commands

The :STATus:OPERation:POWer commands control the Power Operation Register.

Table 159 :STATus:OPERation:POWer Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:OPERation:POWer:BIT:CONDition? (see page 1384)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POWer:BIT:ENABle <bit_value> (see page 1385)	:STATus:OPERation:POWer:BIT:ENABle? (see page 1385)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POWer:BIT:NTRansition <bit_value> (see page 1386)	:STATus:OPERation:POWer:BIT:NTRansition? (see page 1386)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:OPERation:POWer:BIT:PTRansition <bit_value> (see page 1387)	:STATus:OPERation:POWer:BIT:PTRansition? (see page 1387)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POWer:BIT[:EVENT]? (see page 1388)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POWer:CONDition? (see page 1389)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:POWer:ENABle <sum_of_bits> (see page 1390)	:STATus:OPERation:POWer:ENABle? (see page 1390)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:OPERation:POWer:NTRansition <sum_of_bits> (see page 1391)	:STATus:OPERation:POWer:NTRansition? (see page 1391)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 159 :STATus:OPERation:POWer Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:OPERation:POWer:PTRansition <sum_of_bits> (see page 1392)	:STATus:OPERation:POWer:PTRansition? (see page 1392)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:OPERation:POWer[:EVENTt]? (see page 1393)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Power Operation Register To see where the Power Operation Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Power Operation Register.

Table 160 Power Operation Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	Deskew Complete	Deskew Complete	A power application deskew operation is complete.
1	Apply Complete	Apply Complete	A power application apply operation is complete.
0	Setup Complete	Setup Complete	A power application setup operation is complete.

When enabled, the summary message for the Power Operation Register goes to Operation Status Register bit 7 (Power). See **":STATus:OPERation Commands"** on page 1288.

:STATus:OPERation:POWer:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:POWer:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:POWer:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:POWer:BIT:ENABLE"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABLE"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:BIT:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:BIT:ENABle <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:POWer:BIT:ENABle command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:POWer:BIT:ENABle?

The :STATus:OPERation:POWer:BIT:ENABle? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABle"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:POWer:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:POWer:BIT:NTRansition?

The :STATus:OPERation:POWer:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABle"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABle"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:OPERation:POWer:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:OPERation:POWer:BIT:PTRansition?

The :STATus:OPERation:POWer:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABle"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABle"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:POWer:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:OPERation:POWer:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABle"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABle"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:POWer:CONDition?

The :STATus:OPERation:POWer:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABLE"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:ENABLE"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:ENABle

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:POWer:ENABle command sets bits in the Enable register that identify which Event register bits are ORED to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:OPERation:POWer:ENABle?

The :STATus:OPERation:POWer:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABle"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:POWer:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:OPERation:POWer:NTRansition?

The :STATus:OPERation:POWer:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABLE"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABLE"](#) on page 1390
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:OPERation:POWer:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:OPERation:POWer:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:OPERation:POWer:PTRansition?

The :STATus:OPERation:POWer:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABLE"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABLE"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer\[:EVENT\]?"](#) on page 1393
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:OPERation:POWer[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:OPERation:POWer[:EVENT]?

The :STATus:OPERation:POWer[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:OPERation:POWer:BIT:CONDition?"](#) on page 1384
 - [":STATus:OPERation:POWer:BIT:ENABle"](#) on page 1385
 - [":STATus:OPERation:POWer:BIT:NTRansition"](#) on page 1386
 - [":STATus:OPERation:POWer:BIT:PTRansition"](#) on page 1387
 - [":STATus:OPERation:POWer:BIT\[:EVENT\]?"](#) on page 1388
 - [":STATus:OPERation:POWer:CONDition?"](#) on page 1389
 - [":STATus:OPERation:POWer:ENABle"](#) on page 1390
 - [":STATus:OPERation:POWer:NTRansition"](#) on page 1391
 - [":STATus:OPERation:POWer:PTRansition"](#) on page 1392
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger Commands

The :STATus:TRIGger commands control the Trigger Event Register.

Table 161 :STATus:TRIGger Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:TRIGger:BIT:CONDition? (see page 1397)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:ENABle <bit_value> (see page 1398)	:STATus:TRIGger:BIT:ENABle? (see page 1398)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:NTRansition <bit_value> (see page 1399)	:STATus:TRIGger:BIT:NTRansition? (see page 1399)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:TRIGger:BIT:PTRansition <bit_value> (see page 1400)	:STATus:TRIGger:BIT:PTRansition? (see page 1400)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger:BIT[:EVENT]? (see page 1401)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger:CONDition? (see page 1402)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:TRIGger:ENABle <sum_of_bits> (see page 1403)	:STATus:TRIGger:ENABle? (see page 1403)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:TRIGger:NTRansition <sum_of_bits> (see page 1404)	:STATus:TRIGger:NTRansition? (see page 1404)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 161 :STATus:TRIGger Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:TRIGger:PTRan sition <sum_of_bits> (see page 1405)	:STATus:TRIGger:PTRan sition? (see page 1405)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:TRIGger[:EVEN t]? (see page 1406)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Trigger Event Register To see where the Trigger Event Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the Trigger Event Register.

Table 162 Trigger Event Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	---	---	(Not used.)
1	---	---	(Not used.)
0	TER	Trigger Event	A trigger has occurred.

When enabled, the summary message for the Trigger Event Register goes to Status Byte Register bit 0 (TRG). See "***CLS (Clear Status)**" on page 236.

:STATus:TRIGger:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:TRIGger:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:TRIGger:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABle"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:TRIGger:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:TRIGger:BIT:ENABLE?

The :STATus:TRIGger:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABLE"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:TRIGger:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:TRIGger:BIT:NTRansition?

The :STATus:TRIGger:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABle"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:TRIGger:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:TRIGger:BIT:PTRansition?

The :STATus:TRIGger:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABle"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:TRIGger:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATUS:TRIGger:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATUS:TRIGger:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

NOTE

The :TER? query is an alias for the :STATUS:TRIGger:BIT0[:EVENT]? query.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATUS:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATUS:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATUS:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATUS:TRIGger:CONDition?"](#) on page 1402
 - [":STATUS:TRIGger:ENABle"](#) on page 1403
 - [":STATUS:TRIGger:NTRansition"](#) on page 1404
 - [":STATUS:TRIGger:PTRansition"](#) on page 1405
 - [":STATUS:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":TER? \(Trigger Event Register\)"](#) on page 280

:STATus:TRIGger:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:TRIGger:CONDition?

The :STATus:TRIGger:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:ENABle"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:ENABle

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:TRIGger:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:TRIGger:ENABle?

The :STATus:TRIGger:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:TRIGger:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:TRIGger:NTRansition?

The :STATus:TRIGger:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABLE"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABLE"](#) on page 1403
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:TRIGger:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:TRIGger:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:TRIGger:PTRansition?

The :STATus:TRIGger:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABLE"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABLE"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger\[:EVENT\]?"](#) on page 1406
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:TRIGger[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:TRIGger[:EVENT]?

The :STATus:TRIGger[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

NOTE

Because bit 0 is the only bit being used in the Trigger Event Register and the only one that can be latched into the Event register, the :TER? query is an alias for the :STATus:TRIGger[:EVENT]? query.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:TRIGger:BIT:CONDition?"](#) on page 1397
 - [":STATus:TRIGger:BIT:ENABle"](#) on page 1398
 - [":STATus:TRIGger:BIT:NTRansition"](#) on page 1399
 - [":STATus:TRIGger:BIT:PTRansition"](#) on page 1400
 - [":STATus:TRIGger:BIT\[:EVENT\]?"](#) on page 1401
 - [":STATus:TRIGger:CONDition?"](#) on page 1402
 - [":STATus:TRIGger:ENABle"](#) on page 1403
 - [":STATus:TRIGger:NTRansition"](#) on page 1404
 - [":STATus:TRIGger:PTRansition"](#) on page 1405
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":TER? \(Trigger Event Register\)"](#) on page 280

:STATus:USER Commands

The :STATus:USER commands control the User Event Register.

Table 163 :STATus:USER Commands Summary

Command	Query	Options and Query Returns
n/a	:STATus:USER:BIT:CONDition? (see page 1410)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:ENABLE <bit_value> (see page 1411)	:STATus:USER:BIT:ENABLE? (see page 1411)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:NTRansition <bit_value> (see page 1412)	:STATus:USER:BIT:NTRansition? (see page 1412)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
:STATus:USER:BIT:PTRansition <bit_value> (see page 1413)	:STATus:USER:BIT:PTRansition? (see page 1413)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:USER:BIT[:EVENT]? (see page 1414)	<bit_value> ::= {0 1 } ::= 0-14; an integer in NR1 format
n/a	:STATus:USER:CONDition? (see page 1415)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:USER:ENABLE <sum_of_bits> (see page 1416)	:STATus:USER:ENABLE? (see page 1416)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
:STATus:USER:NTRansition <sum_of_bits> (see page 1417)	:STATus:USER:NTRansition? (see page 1417)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

Table 163 :STATus:USER Commands Summary (continued)

Command	Query	Options and Query Returns
:STATus:USER:PTRansition <sum_of_bits> (see page 1418)	:STATus:USER:PTRansition? (see page 1418)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format
n/a	:STATus:USER[:EVENT]? (see page 1419)	<sum_of_bits> ::= 0-32767; an integer in NR1 format ::= 0-14; an integer in NR1 format

User Event Register To see where the User Event Register appears in the status register hierarchy, see [Chapter 46](#), “Status Reporting,” starting on page 1669.

To understand the SCPI status register model, see ["Introduction to :STATus Commands"](#) on page 1283.

The following bits appear in the User Event Register.

Table 164 User Event Register Bits

Bit	Name	Description	When Set (1 = High = True), Indicates:
15	---	---	(Not used, always zero.)
14	---	---	(Not used.)
13	---	---	(Not used.)
12	---	---	(Not used.)
11	---	---	(Not used.)
10	---	---	(Not used.)
9	---	---	(Not used.)
8	---	---	(Not used.)
7	---	---	(Not used.)
6	---	---	(Not used.)
5	---	---	(Not used.)
4	---	---	(Not used.)
3	---	---	(Not used.)
2	---	---	(Not used.)
1	---	---	(Not used.)
0	USR	User Event	An enabled user event condition has occurred.

When enabled, the summary message for the Trigger Event Register goes to Status Byte Register bit 1 (USR). See "***CLS (Clear Status)**" on page 236.

:STATus:USER:BIT:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:USER:BIT:CONDition?

 ::= 0-14; an integer in NR1 format

The :STATus:USER:BIT:CONDition? query returns the value of a particular bit in the Condition register.

The Condition register continuously monitors status. Reading a Condition register bit does not affect its contents.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABle"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATUS:USER:BIT:ENABLE

N (see [page 1708](#))

Command Syntax :STATUS:USER:BIT:ENABLE <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATUS:USER:BIT:ENABLE command sets a particular bit in the Enable register that identifies a corresponding Event register bit that will be ORed with other identified Event register bits to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATUS:USER:BIT:ENABLE?

The :STATUS:USER:BIT:ENABLE? query returns the value of the specified Enable register bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATUS:USER:BIT:CONDition?"](#) on page 1410
 - [":STATUS:USER:BIT:NTRansition"](#) on page 1412
 - [":STATUS:USER:BIT:PTRansition"](#) on page 1413
 - [":STATUS:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATUS:USER:CONDition?"](#) on page 1415
 - [":STATUS:USER:ENABLE"](#) on page 1416
 - [":STATUS:USER:NTRansition"](#) on page 1417
 - [":STATUS:USER:PTRansition"](#) on page 1418
 - [":STATUS:USER\[:EVENT\]?"](#) on page 1419
 - [":STATUS:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:BIT:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:USER:BIT:NTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:USER:BIT:NTRansition command sets a particular bit in the Negative Transition filter that identifies a Condition register bit on which a transition from 1 to 0 or TRUE to FALSE will be latched into the corresponding Event register bit.

Query Syntax :STATus:USER:BIT:NTRansition?

The :STATus:USER:BIT:NTRansition? query returns the value of the specified Negative Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABle"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:BIT:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:USER:BIT:PTRansition <bit_value>

 ::= 0-14; an integer in NR1 format

<bit_value> ::= {0 | 1}

The :STATus:USER:BIT:PTRansition command sets a particular bit in the Positive Transition filter that identifies a Condition register bit on which a transition from 0 to 1 or FALSE to TRUE will be latched into the corresponding Event register bit.

Query Syntax :STATus:USER:BIT:PTRansition?

The :STATus:USER:BIT:PTRansition? query returns the value of the specified Positive Transition filter bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABLE"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABLE"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:BIT[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:USER:BIT[:EVENT]?

 ::= 0-14; an integer in NR1 format

The :STATus:USER:BIT[:EVENT]? query returns the value of the specified Event register bit and clears the bit.

Return Format <bit_value><NL>

<bit_value> ::= {0 | 1}

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABle"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:CONDition?

N (see [page 1708](#))

Query Syntax :STATus:USER:CONDition?

The :STATus:USER:CONDition? query returns the decimal value of the sum of the bits in the Condition register.

The Condition register continuously monitors status. Reading the Condition register does not affect its contents.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:ENABle"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:ENABle

N (see [page 1708](#))

Command Syntax :STATus:USER:ENABle <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:USER:ENABle command sets bits in the Enable register that identify which Event register bits are ORed to create the Summary Message bit that is sent to the parent status register.

Query Syntax :STATus:USER:ENABle?

The :STATus:USER:ENABle? query returns the decimal value of the sum of the bits in the Enable register.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:NTRansition

N (see [page 1708](#))

Command Syntax :STATus:USER:NTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:USER:NTRansition command sets bits in the Negative Transition filter that identify the Condition register bits on which transitions from 1 to 0 or TRUE to FALSE will be latched into the Event register.

Query Syntax :STATus:USER:NTRansition?

The :STATus:USER:NTRansition? query returns the decimal value of the sum of the bits in the Negative Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABLE"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABLE"](#) on page 1416
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER:PTRansition

N (see [page 1708](#))

Command Syntax :STATus:USER:PTRansition <sum_of_bits>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

The :STATus:USER:PTRansition command sets bits in the Positive Transition filter that identify the Condition register bits on which transitions from 0 to 1 or FALSE to TRUE will be latched into the Event register.

Query Syntax :STATus:USER:PTRansition?

The :STATus:USER:PTRansition? query returns the decimal value of the sum of the bits in the Positive Transition filter.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABLE"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABLE"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER\[:EVENT\]?"](#) on page 1419
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

:STATus:USER[:EVENT]?

N (see [page 1708](#))

Query Syntax :STATus:USER[:EVENT]?

The :STATus:USER[:EVENT]? query returns the decimal value of the sum of the bits in the Event register and clears the register.

The contents of the Event register show the events that have occurred since the last time the register was read.

Return Format <sum_of_bits><NL>

<sum_of_bits> ::= 0-32767; an integer in NR1 format

- See Also**
- [":STATus:USER:BIT:CONDition?"](#) on page 1410
 - [":STATus:USER:BIT:ENABle"](#) on page 1411
 - [":STATus:USER:BIT:NTRansition"](#) on page 1412
 - [":STATus:USER:BIT:PTRansition"](#) on page 1413
 - [":STATus:USER:BIT\[:EVENT\]?"](#) on page 1414
 - [":STATus:USER:CONDition?"](#) on page 1415
 - [":STATus:USER:ENABle"](#) on page 1416
 - [":STATus:USER:NTRansition"](#) on page 1417
 - [":STATus:USER:PTRansition"](#) on page 1418
 - [":STATus:PRESet"](#) on page 1287
 - ["*CLS \(Clear Status\)"](#) on page 236

38 :SYSTem Commands

Control basic system functions of the oscilloscope. See ["Introduction to :SYSTem Commands"](#) on page 1424.

Table 165 :SYSTem Commands Summary

Command	Query	Options and Query Returns
:SYSTem:DATE <date> (see page 1425)	:SYSTem:DATE? (see page 1425)	<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}
n/a	:SYSTem:DIDentifier? (see page 1426)	n/a
:SYSTem:DSP <string> (see page 1427)	n/a	<string> ::= up to 75 characters as a quoted ASCII string
n/a	:SYSTem:ERRor:ALL? (see page 1428)	<errors_list> ::= comma-separated list of all errors from the queue (codes and strings)
n/a	:SYSTem:ERRor:CODE[:N EXT]? (see page 1429)	<error_code> ::= the next error (code only) from the queue
n/a	:SYSTem:ERRor:COUNt? (see page 1430)	<count> ::= an integer number of errors in the queue
n/a	:SYSTem:ERRor:EXTended? <type> (see page 1431)	<type> ::= {INFO WARNING ERROR FATAL MESSAGE ALL} <errors_list> ::= returns a comma-separated list of all errors (strings, no codes) of the specified type from the queue

Table 165 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:SYSTem:ERRor[:NEXT]? (see page 1432)	<error_number>, <error_string> ::= the next error (code and string) from the queue <error_number> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see page 1661).
n/a	:SYSTem:HELP:HEADers? (see page 1434)	<binary_block> ::= newline-separated list of supported SCPI commands/queries
n/a	:SYSTem:HID? (see page 1435)	n/a
:SYSTem:LOCK <value> (see page 1436)	:SYSTem:LOCK? (see page 1436)	<value> ::= {{1 ON} {0 OFF}}
n/a	:SYSTem:LOCK:OWNer? (see page 1437)	<session_string> ::= quoted session string or "NONE"
:SYSTem:LOCK:RELease (see page 1438)	n/a	n/a
n/a	:SYSTem:LOCK:REQuest? (see page 1439)	{1 0} (for granted or not)
:SYSTem:PERSon[:MANu facturer] <manufacturer_string> (see page 1440)	:SYSTem:PERSon[:MANu facturer]? (see page 1440)	<manufacturer_string> ::= quoted ASCII string, up to 63 characters
:SYSTem:PERSon[:MANu facturer]:DEFault (see page 1441)	n/a	Sets manufacturer string to "KEYSIGHT TECHNOLOGIES"
:SYSTem:POWercycle (see page 1442)	n/a	n/a
:SYSTem:PRECision:LEN Gth <length> (see page 1443)	:SYSTem:PRECision:LEN Gth? (see page 1443)	<length> ::= {64K 128K 256K 512K 1M 1.5M 2M 4M 8M 12M 16M 24M 32M}
:SYSTem:PRECision:PEA K {{0 OFF} {1 ON}} (see page 1444)	:SYSTem:PRECision:PEA K? (see page 1444)	<setting> ::= {0 1}
:SYSTem:PRESet (see page 1445)	n/a	See :SYSTem:PRESet (see page 1445)

Table 165 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:PRODUCTION:FR ESet (see page 1445)	n/a	See :SYSTem:PRODUCTION:FRESet (see page 1448)
:SYSTem:PROTECTION:LO CK <value> (see page 1449)	:SYSTem:PROTECTION:LO CK? (see page 1449)	<value> ::= {{1 ON} {0 OFF}}
:SYSTem:RLOGger <setting>[,<file_name >[,<write_mode>]] (see page 1450)	n/a	<setting> ::= {{0 OFF} {1 ON}} <file_name> ::= quoted ASCII string <write_mode> ::= {CREate APPend}
:SYSTem:RLOGger:DESTi nation <dest> (see page 1451)	:SYSTem:RLOGger:DESTi nation? (see page 1451)	<dest> ::= {FILE SCReen BOTH}
:SYSTem:RLOGger:DISPl ay {{0 OFF} {1 ON}} (see page 1452)	:SYSTem:RLOGger:DISPl ay? (see page 1452)	<setting> ::= {0 1}
:SYSTem:RLOGger:FNAME <file_name> (see page 1453)	:SYSTem:RLOGger:FNAME ? (see page 1453)	<file_name> ::= quoted ASCII string
:SYSTem:RLOGger:STATe {{0 OFF} {1 ON}} (see page 1454)	:SYSTem:RLOGger:STATe ? (see page 1454)	<setting> ::= {0 1}
:SYSTem:RLOGger:TRANs parent {{0 OFF} {1 ON}} (see page 1455)	:SYSTem:RLOGger:TRANs parent? (see page 1455)	<setting> ::= {0 1}
:SYSTem:RLOGger:WMODE <write_mode> (see page 1456)	:SYSTem:RLOGger:WMODE ? (see page 1456)	<write_mode> ::= {CREate APPend}
:SYSTem:SETup <setup_data> (see page 1457)	:SYSTem:SETup? (see page 1457)	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:SHUTdown (see page 1459)	n/a	n/a
:SYSTem:TIME <time> (see page 1460)	:SYSTem:TIME? (see page 1460)	<time> ::= hours,minutes,seconds in NR1 format
n/a	:SYSTem:TIME:DSAVing? (see page 1461)	<setting> ::= {0 1}

Table 165 :SYSTem Commands Summary (continued)

Command	Query	Options and Query Returns
:SYSTem:TIME:NTProTol {{0 OFF} {1 ON}} (see page 1462)	:SYSTem:TIME:NTProTol? (see page 1462)	<setting> ::= {0 1}
:SYSTem:TIME:ZONE <zone> (see page 1463)	:SYSTem:TIME:ZONE? (see page 1463)	<zone> ::= {UP0000 Europe_London UP0100 Europe_Paris UP0200 Africa_Cairo UP0300 UP0330 Asia_Tehran UP0400 Asia_Dubai UP0430 Asia_Kabul UP0500 Asia_Karachi UP0530 Asia_Kolkata UP0600 Asia_Dhaka UP0700 Asia_Bangkok UP0800 Asia_Hong_Kong UP0845 Australia_Eucla UP0900 Asia_Tokyo UP0930 Australia_Adelaide Australia_Darwin UP1000 Australia_Brisbane Australia_Sydney UP1030 Australia_Lord_Howe UP1100 Pacific_Noumea UP1200 UM0100 UM0200 UM0300 America_Sao_Paulo UM0330 America_St_Johns UM0400 America_Caracas UM0500 America_New_York UM0600 America_Chicago UM0700 America_Denver UM0800 America_Los_Angeles UM0900 America_Anchorage UM1000 Pacific_Honolulu UM1100 UM1200}
:SYSTem:TOUCh {{1 ON} {0 OFF}} (see page 1464)	:SYSTem:TOUCh? (see page 1464)	{1 0}
n/a	:SYSTem:VERSion? (see page 1465)	<SCPI_version_implemented> ::= 1999.0

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 1708](#))

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 1424
 - [":SYSTem:TIME"](#) on page 1460

:SYSTem:DIIdentifier?

N (see [page 1708](#))

Query Syntax :SYSTem:DIIdentifier?

The :SYSTem:DIIdentifier? query returns the oscilloscope's Host ID string.

The oscilloscope's Host ID is needed when redeeming licenses for oscilloscope upgrades or other licensed features.

The exact format of returned string are product-specific. This example returns a model number and serial number in a comma-separated format:

```
X12345A,US12345678
```

Portable programs should not attempt to parse the contents of the returned string. Use the *IDN? query instead to get the model number and/or serial number in a defined portable format.

NOTE

This query is the same as the :SYSTem:HID? query.

Return Format <host_id><NL>

```
<host_id> ::= ASCII string
```

- See Also**
- ["*IDN? \(Identification Number\)"](#) on page 241
 - [":SYSTem:HID?"](#) on page 1435
 - ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

:SYSTem:DSP

N (see [page 1708](#))

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 and save it in the message log. (Two sets of quote marks without a space between them creates a NULL string.)

Also, to manually remove the message from the display and save it in the message log, you can:

- Press any front panel key.
- Select the close button ("X") in the graphical user interface's message dialog box.

See Also • ["Introduction to :SYSTem Commands"](#) on page 1424

:SYSTem:ERRor:ALL?

C (see [page 1708](#))

Query Syntax :SYSTem:ERRor:ALL?

The :SYSTem:ERRor:ALL? query returns a comma-separated list of all errors from the queue (codes and strings).

Return Format <errors_list><NL>

<errors_list> ::= comma-separated list of all errors from the queue
(codes and strings)

- See Also**
- [":SYSTem:ERRor:CODE\[:NEXT\]?"](#) on page 1429
 - [":SYSTem:ERRor:COUNt?"](#) on page 1430
 - [":SYSTem:ERRor:EXTended?"](#) on page 1431
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432

:SYSTem:ERRor:CODE[:NEXT]?

C (see [page 1708](#))

Query Syntax :SYSTem:ERRor:CODE[:NEXT] ?

The :SYSTem:ERRor:CODE[:NEXT]? query returns the next error (code only) from the queue.

Except for the shortened response, the query operates identically to the SYSTem:ERRor[:NEXT]? query.

Return Format <error_code><NL>

<error_code> ::= the next error (code only) from the queue

- See Also**
- [":SYSTem:ERRor:ALL?"](#) on page 1428
 - [":SYSTem:ERRor:COUNT?"](#) on page 1430
 - [":SYSTem:ERRor:EXTended?"](#) on page 1431
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432

:SYSTem:ERRor:COUNT?

C (see [page 1708](#))

Query Syntax :SYSTem:ERRor:COUNT?

The :SYSTem:ERRor:COUNT? query returns an integer number of errors in the queue.

Return Format <count><NL>

<count> ::= an integer number of errors in the queue

- See Also**
- [":SYSTem:ERRor:ALL?"](#) on page 1428
 - [":SYSTem:ERRor:CODE\[:NEXT\]?"](#) on page 1429
 - [":SYSTem:ERRor:EXTended?"](#) on page 1431
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432

:SYSTem:ERRor:EXTended?

C (see [page 1708](#))

Query Syntax :SYSTem:ERRor:EXTended? <type>

<type> ::= {INFO | WARNING | ERROR | FATAL | MESSAGE | ALL}

The :SYSTem:ERRor:EXTended? query returns a comma-separated list of all errors (strings, no codes) of the specified type from the queue.

Return Format <errors_list><NL>

<errors_list> ::= comma-separated list of all errors (strings, no codes) of the specified type from the queue

- See Also**
- [":SYSTem:ERRor:ALL?"](#) on page 1428
 - [":SYSTem:ERRor:CODE\[:NEXT\]?"](#) on page 1429
 - [":SYSTem:ERRor:COUNt?"](#) on page 1430
 - [":SYSTem:ERRor\[:NEXT\]?"](#) on page 1432

:SYSTem:ERRor[:NEXT]?

C (see [page 1708](#))

Query Syntax `:SYSTem:ERRor[:NEXT] ?`

The `:SYSTem:ERRor[:NEXT]?` 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[:NEXT]?` 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.

Error Queue 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).

When the error queue is read with the `:SYSTem:ERRor[:NEXT]?` query, the oldest error is removed 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.

Return Format `<error_number>,<error_string><NL>`

`<error_number>` ::= an integer error code in NR1 format

`<error_string>` ::= quoted ASCII string containing the error message

Error messages are listed in [Chapter 45](#), "Error Messages," starting on page 1661.

- See Also**
- ["Introduction to :SYSTem Commands"](#) on page 1424
 - ["*ESR? \(Standard Event Status Register\)"](#) on page 239
 - ["*CLS \(Clear Status\)"](#) on page 236
 - [":SYSTem:ERRor:ALL?"](#) on page 1428
 - [":SYSTem:ERRor:CODE\[:NEXT\]?"](#) on page 1429

- **":SYSTem:ERRor:COUNT?"** on page 1430
- **":SYSTem:ERRor:EXTended?"** on page 1431

:SYSTem:HELP:HEADers?

N (see [page 1708](#))

Query Syntax :SYSTem:HELP:HEADers?

The :SYSTem:HELP:HEADers? query returns a binary block that contains a newline-separated list of supported SCPI commands and queries.

Return Format <binary_block><NL>

<binary_block> ::= newline-separated list of supported SCPI commands/queries

Lines can have the following suffixes:

- /nquery/ – There is no "query" version of the SCPI header.
- /qonly/ – There is no "command" version of the SCPI header.

For example, a portion of the returned result looks like:

```
...
*SRE
*STB?/qonly/
*TRG/nquery/
*TST?/qonly/
*WAI/nquery/
:ABORt/nquery/
:ACQuire:COMPlEte
:ACQuire:COUNt
:ACQuire:DIGitizer
:ACQuire:MANual
:ACQuire:MODE
:ACQuire:POINts[:ANALog]
:ACQuire:POINts[:ANALog]:AUTO
:ACQuire:RSIGnal
...
```

Square brackets in a SCPI header show optional parts of a command/query. The previous example shows :ACQuire:POINts:AUTO is the same as :ACQuire:POINts:ANALog:AUTO.

:SYSTem:HID?

N (see [page 1708](#))

Query Syntax :SYSTem:HID?

The :SYSTem:HID? query returns the oscilloscope's Host ID string.

The oscilloscope's Host ID is needed when redeeming licenses for oscilloscope upgrades or other licensed features.

The exact format of returned string are product-specific. This example returns a model number and serial number in a comma-separated format:

```
X12345A,US12345678
```

Portable programs should not attempt to parse the contents of the returned string. Use the *IDN? query instead to get the model number and/or serial number in a defined portable format.

NOTE

This query is the same as the :SYSTem:DIentifier? query.

Return Format <host_id><NL>

```
<host_id> ::= ASCII string
```

- See Also**
- ["*IDN? \(Identification Number\)"](#) on page 241
 - [":SYSTem:DIentifier?"](#) on page 1426
 - ["Queries Not Supported in Multiple Program Message Units"](#) on page 1713

:SYSTem:LOCK

N (see [page 1708](#))

Command Syntax :SYSTem:LOCK <on_off>

<on_off> ::= {{1 | ON} | {0 | OFF}}

The :SYSTem:LOCK command disables the front panel (and touch screen). 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 1424

:SYSTem:LOCK:OWNer?

N (see [page 1708](#))

Query Syntax :SYSTem:LOCK:OWNer?

The :SYSTem:LOCK:OWNer? query returns what session currently has the lock on this device. If no session has a lock then "NONE" is returned.

Return Format <session_string><NL>
 <session_string> ::= quoted session string or "NONE"

- See Also**
- [":SYSTem:LOCK:RELease"](#) on page 1438
 - [":SYSTem:LOCK:REQuest?"](#) on page 1439

:SYSTem:LOCK:RELease

N (see [page 1708](#))

Command Syntax :SYSTem:LOCK:RELease

The :SYSTem:LOCK:RELease command releases the lock if owned by this session.

If this session does not have the lock, this command has no effect.

- See Also**
- [":SYSTem:LOCK:OWNer?"](#) on page 1437
 - [":SYSTem:LOCK:REQuest?"](#) on page 1439

:SYSTem:LOCK:REQuest?

N (see [page 1708](#))

Query Syntax :SYSTem:LOCK:REQuest?

The :SYSTem:LOCK:REQuest? query attempts to attain the lock on this device and returns 1 if successful or 0 if it fails.

Return Format <grant_status><NL>
<grant_status> ::= {1 | 0}

- See Also**
- [":SYSTem:LOCK:OWNer?"](#) on page 1437
 - [":SYSTem:LOCK:RELease"](#) on page 1438

:SYSTem:PERSONa[:MANufacturer]

N (see [page 1708](#))

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 241
 - [":SYSTem:PERSONa\[:MANufacturer\]:DEFAULT"](#) on page 1441
 - ["Introduction to :SYSTem Commands"](#) on page 1424

:SYSTem:PERSonA[:MANufacturer]:DEFault

N (see [page 1708](#))

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 241
 - [":SYSTem:PERSonA\[:MANufacturer\]"](#) on page 1440
 - ["Introduction to :SYSTem Commands"](#) on page 1424

:SYSTem:POWercycle

N (see [page 1708](#))

Command Syntax :SYSTem:POWercycle

The :SYSTem:POWercycle command powers down the oscilloscope and restarts it. This is the same as pressing the front panel power button to power down the oscilloscope and then pressing the power button again to start the oscilloscope.

See Also • [":SYSTem:SHUTdown"](#) on page 1459

:SYSTem:PRECision:LENGth

N (see [page 1708](#))

Command Syntax :SYSTem:PRECision:LENGth <length>

<length> ::= {64K | 128K | 256K | 512K | 1M | 1.5M | 2M | 4M | 8M
| 12M | 16M | 24M | 32M}

The :SYSTem:PRECision:LENGth command specifies the length of the analysis record. You can specify 64K, 128K, 256K, 512K, 1M, 1.5M, 2M, 4M, 8M, 12M, 16M, 24M, or 32M points unless limited by the oscilloscope's overall memory depth.

Query Syntax :SYSTem:PRECision:LENGth?

The :SYSTem:PRECision:LENGth? query returns the current analysis record length setting.

Return Format <length><NL>

<length> ::= {64K | 128K | 256K | 512K | 1M | 1.5M | 2M | 4M | 8M
| 12M | 16M | 24M | 32M}

- See Also**
- ["Introduction to :SYSTem Commands"](#) on page 1424
 - [":SYSTem:PRECision:PEAK"](#) on page 1444
 - [":WAVEform:POINts:MODE"](#) on page 1568
 - ["*RST \(Reset\)"](#) on page 247

:SYSTem:PRECision:PEAK

N (see [page 1708](#))

Command Syntax :SYSTem:PRECision:PEAK {{0 | OFF} | {1 | ON}}

The :SYSTem:PRECision:PEAK command enables or disables the **Analyze Peaks** option for the analysis record. When enabled, peak detection is used when decimating the raw acquisition record into the *analysis record*. For each block of samples that must be reduced, the maximum and minimum samples are kept. When disabled, the decimator is configured to provide a best-estimate of all the samples that each point in the record represents, with no filtering.

Query Syntax :SYSTem:PRECision:PEAK?

The :SYSTem:PRECision:PEAKs? query returns the **Analyze Peaks** option setting.

Return Format <setting><NL>
<setting> ::= {0 | 1}

See Also • [":SYSTem:PRECision:LENGth"](#) on page 1443

:SYSTem:PRESet

C (see [page 1708](#))

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 selecting **Control > Default Setup** in the graphical user interface.

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 Settings	
Mode	Normal
Averaging	Off
# Averages	8

Analog Channel Settings	
Channel 1	On
Channel 2/3/4	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 Settings	
Source	Channel 1

Digital Channel Settings	
Channel 0 - 7	Off
Labels	Off
Threshold	TTL (1.4 V)

Display Settings	
Persistence	Off
Grid	20%

Measurement Settings	
Source	Channel 1

Run Control Settings	
	Scope is running

Time Base Settings	
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

Trigger Settings	
Type	Edge
Mode	Auto
Coupling	dc
Source	Channel 1
Level	0.0 V
Slope	Positive

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

- See Also
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*RST \(Reset\)"](#) on page 247

:SYSTem:PRODUCTION:FRESet

C (see [page 1708](#))

Command Syntax :SYSTem:PRODUCTION:FRESet

The :SYSTem:PRODUCTION:FRESet command wipes the oscilloscope's non-volatile storage like **Secure Erase**, but it does not perform the National Industrial Security Program Operation Manual (NISPOM) Chapter 8 procedure and is much faster.

- See Also**
- ["Introduction to Common \(*\) Commands"](#) on page 235
 - ["*RST \(Reset\)"](#) on page 247
 - [":SYSTem:PRESet"](#) on page 1445

:SYSTem:PROTection:LOCK

N (see [page 1708](#))

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.

NOTE

Be careful when turning ON the :SYSTem:PROTection:LOCK because there is no visual indication of this setting in the front-panel user interface (other than a disabled 50 Ω channel input impedance selection), and a user could think the oscilloscope is not working properly.

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 1424

:SYSTem:RLOGger

N (see [page 1708](#))

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 1451
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:STATE"](#) on page 1454
 - [":SYSTem:RLOGger:TRANSPARENT"](#) on page 1455
 - [":SYSTem:RLOGger:WMODE"](#) on page 1456

:SYSTem:RLOGger:DESTination

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:STAtE"](#) on page 1454
 - [":SYSTem:RLOGger:TRANSPARENT"](#) on page 1455
 - [":SYSTem:RLOGger:WMODe"](#) on page 1456

:SYSTem:RLOGger:DISPlay

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DESTination"](#) on page 1451
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:STATE"](#) on page 1454
 - [":SYSTem:RLOGger:TRANSPARENT"](#) on page 1455
 - [":SYSTem:RLOGger:WMODE"](#) on page 1456

:SYSTem:RLOGger:FNAME

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DESTination"](#) on page 1451
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:STAtE"](#) on page 1454
 - [":SYSTem:RLOGger:TRANSPARENT"](#) on page 1455
 - [":SYSTem:RLOGger:WMODe"](#) on page 1456

:SYSTem:RLOGger:STATe

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DESTination"](#) on page 1451
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:TRANsparent"](#) on page 1455
 - [":SYSTem:RLOGger:WMODE"](#) on page 1456

:SYSTem:RLOGger:TRANSPARENT

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DESTination"](#) on page 1451
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:STaTe"](#) on page 1454
 - [":SYSTem:RLOGger:WMODE"](#) on page 1456

:SYSTem:RLOGger:WMODE

N (see [page 1708](#))

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 1450
 - [":SYSTem:RLOGger:DESTination"](#) on page 1451
 - [":SYSTem:RLOGger:DISPlay"](#) on page 1452
 - [":SYSTem:RLOGger:FNAME"](#) on page 1453
 - [":SYSTem:RLOGger:STAtE"](#) on page 1454
 - [":SYSTem:RLOGger:TRANSPARENT"](#) on page 1455

:SYSTem:SETup

C (see [page 1708](#))

Command Syntax :SYSTem:SETup <setup_data>

<setup_data> ::= binary block data in IEEE 488.2 # format.

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.

Query Syntax :SYSTem:SETup?

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.

Return Format <setup_data><NL>

<setup_data> ::= binary block data in IEEE 488.2 # format

- See Also**
- ["Introduction to :SYSTem Commands"](#) on page 1424
 - ["*LRN? \(Learn Device Setup\)"](#) on page 242

Example Code

```
' 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<setup string><NL>
' 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.scp"

' Open file for output.
Dim hFile As Integer
hFile = FreeFile
Open strPath For Binary Access Write Lock Write As hFile
Put hFile, , varQueryResult ' Write data.
Close hFile ' 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.scp"

' Open file for input.
Open strPath For Binary Access Read As hFile
Get hFile, , varSetupString ' Read data.
Close hFile ' Close file.
```

```
' Write setup string back to oscilloscope using ":SYSTem:SETup"  
' command:  
myScope.WriteIEEEBlock ":SYSTem:SETup ", varSetupString  
CheckForInstrumentErrors
```

See complete example programs at: [Chapter 49](#), “Programming Examples,” starting on page 1717

:SYSTem:SHUTdown

N (see [page 1708](#))

Command Syntax :SYSTem:SHUTdown

The :SYSTem:SHUTdown command powers down the oscilloscope. This is the same as pressing the front panel power button to power down the oscilloscope. This allows the oscilloscope to shut down gracefully.

See Also • [":SYSTem:POWercycle"](#) on page 1442

:SYSTem:TIME

N (see [page 1708](#))

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 1424
 - [":SYSTem:DATE"](#) on page 1425
 - [":SYSTem:TIME:DSAVing?"](#) on page 1461
 - [":SYSTem:TIME:NTPRotocol"](#) on page 1462
 - [":SYSTem:TIME:ZONE"](#) on page 1463

:SYSTem:TIME:DSAVing?

N (see [page 1708](#))

Query Syntax :SYSTem:TIME:DSAVing?

For the selected time zone (see :SYSTem:TIME:ZONE), the :SYSTem:TIME:DSAVing? query returns whether "automatically adjust clock for Daylight Savings Time" is enabled or disabled.

Return Format <on_off><NL>

<on_off> ::= {1 | 0}

- See Also**
- ["Introduction to :SYSTem Commands"](#) on page 1424
 - [":SYSTem:TIME"](#) on page 1460
 - [":SYSTem:TIME:NTPRotocol"](#) on page 1462
 - [":SYSTem:TIME:ZONE"](#) on page 1463

:SYSTem:TIME:NTPRotocol

N (see [page 1708](#))

Command Syntax :SYSTem:TIME:NTPRotocol {{0 | OFF} | {1 | ON}}

The :SYSTem:TIME:NTPRotocol command enables or disables automatically setting the oscilloscope's clock using Network Time Protocol.

Query Syntax :SYSTem:TIME:NTPRotocol?

The :SYSTem:TIME:NTPRotocol? query returns whether or not the oscilloscope's clock is being set using Network Time Protocol.

Return Format <setting><NL>
<setting> ::= {0 | 1}

- See Also**
- [":SYSTem:TIME"](#) on page 1460
 - [":SYSTem:TIME:DSAVing?"](#) on page 1461
 - [":SYSTem:TIME:ZONE"](#) on page 1463

:SYSTem:TIME:ZONE

N (see [page 1708](#))

Command Syntax :SYSTem:TIME:ZONE <zone>

```
<zone> ::= {UP0000 | Europe_London | UP0100 | Europe_Paris | UP0200
| Africa_Cairo | UP0300 | UP0330 | Asia_Tehran | UP0400
| Asia_Dubai | UP0430 | Asia_Kabul | UP0500 | Asia_Karachi | UP0530
| Asia_Kolkata | UP0600 | Asia_Dhaka | UP0700 | Asia_Bangkok
| UP0800 | Asia_Hong_Kong | UP0845 | Australia_Eucla | UP0900
| Asia_Tokyo | UP0930 | Australia_Adelaide | Australia_Darwin
| UP1000 | Australia_Brisbane | Australia_Sydney | UP1030
| Australia_Lord_Howe | UP1100 | Pacific_Noumea | UP1200 | UM0100
| UM0200 | UM0300 | America_Sao_Paulo | UM0330 | America_St_Johns
| UM0400 | America_Caracas | UM0500 | America_New_York | UM0600
| America_Chicago | UM0700 | America_Denver | UM0800
| America_Los_Angeles | UM0900 | America_Anchorage | UM1000
| Pacific_Honolulu | UM1100 | UM1200}
```

The :SYSTem:TIME:ZONE command sets the system time zone based on the specified name or offset from UTC.

UTC offsets are: +0, +100, +200, +300, +330, +400, +430, +500, +530, +600, +700, +800, +845, +900, +930, +1000, +1030, +1100, +1200, -100, -200, -300, -330, -400, -500, -600, -700, -800, -900, -1000, -1100, or -1200, where "+" is "UP" and "-" is "UM" followed by the "HHMM" hours and minutes.

The named zones all select the "Automatically adjust clock for Daylight Savings Time" setting (see :SYSTem:TIME:DSaving?). The UTC offset zones may or may not select the "Automatically adjust clock for Daylight Savings Time" setting

Because licenses are based on Coordinated Universal Time (UTC), the proper time zone setting allows licenses to work immediately after they are generated.

Query Syntax :SYSTem:TIME:ZONE?

The :SYSTem:TIME:ZONE? query returns the time zone offset setting.

Return Format <zone><NL>

- See Also**
- [":SYSTem:TIME"](#) on page 1460
 - [":SYSTem:TIME:DSaving?"](#) on page 1461
 - [":SYSTem:TIME:NTPProtocol"](#) on page 1462

:SYSTem:TOUCh

N (see [page 1708](#))

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 1424

:SYSTem:VERSion?

N (see [page 1708](#))

Query Syntax :SYSTem:VERSion?

The :SYSTem:VERSion? query returns the version of the SCPI standard that is implemented by the oscilloscope.

Return Format <SCPI_version_implemented><NL>
<SCPI_version_implemented> ::= 1999.0

39 :TIMebase Commands

Control all horizontal sweep functions. See "[Introduction to :TIMebase Commands](#)" on page 1468.

Table 166 :TIMebase Commands Summary

Command	Query	Options and Query Returns
:TIMebase[:MAIN]:POSITION <pos> (see page 1469)	:TIMebase[:MAIN]:POSITION? (see page 1469)	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMebase[:MAIN]:RANGE <range_value> (see page 1470)	:TIMebase[:MAIN]:RANGE? (see page 1470)	<range_value> ::= time for 10 div in seconds in NR3 format
:TIMebase[:MAIN]:SCALE <scale_value> (see page 1471)	:TIMebase[:MAIN]:SCALE? (see page 1471)	<scale_value> ::= time/div in seconds in NR3 format
:TIMebase:MODE <value> (see page 1472)	:TIMebase:MODE? (see page 1472)	<value> ::= {MAIN WINDOW XY ROLL}
:TIMebase:REFERENCE {LEFT CENTER RIGHT CUSTOM} (see page 1473)	:TIMebase:REFERENCE? (see page 1473)	<return_value> ::= {LEFT CENTER RIGHT CUSTOM}
:TIMebase:REFERENCE:LOCATION <loc> (see page 1474)	:TIMebase:REFERENCE:LOCATION? (see page 1474)	<loc> ::= 0.0 to 1.0 in NR3 format
:TIMebase:VERNier {{0 OFF} {1 ON}} (see page 1475)	:TIMebase:VERNier? (see page 1475)	{0 1}
:TIMebase:WINDOW:POSITION <pos> (see page 1476)	:TIMebase:WINDOW:POSITION? (see page 1476)	<pos> ::= time from the trigger event to the zoomed view reference point in NR3 format

Table 166 :TImebase Commands Summary (continued)

Command	Query	Options and Query Returns
:TImebase:WINDow:RANGe <range_value> (see page 1477)	:TImebase:WINDow:RANGe? (see page 1477)	<range_value> ::= range value in seconds in NR3 format for the zoomed window
:TImebase:WINDow:SCALe <scale_value> (see page 1478)	:TImebase:WINDow:SCALe? (see page 1478)	<scale_value> ::= scale value in seconds in NR3 format for the zoomed window

Introduction to :TImebase Commands The TImebase 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 :TImebase? to query setup information for the TImebase subsystem.

Return Format

The following is a sample response from the :TImebase? query. In this case, the query was issued following a *RST command.

```
:TIM:MODE MAIN;REF CENT;MAIN:RANG +1.000E-03;POS +0.0E+00
```


:TImebase[:MAIN]:POSition

C (see [page 1708](#))

Command Syntax :TImebase[:MAIN]:POSition <pos>

<pos> ::= time in seconds from the trigger to the display reference
in NR3 format

The :TImebase[:MAIN]: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[:MAIN]:DELaY command.

Query Syntax :TImebase[:MAIN]:POSition?

The :TImebase[:MAIN]: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 1468
 - [":TImebase:REFeRence"](#) on page 1473
 - [":TImebase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TImebase\[:MAIN\]:SCALe"](#) on page 1471
 - [":TImebase:WINDow:POSition"](#) on page 1476
 - [":TImebase\[:MAIN\]:DELaY"](#) on page 1659

:TIMEbase[:MAIN]:RANGe

C (see [page 1708](#))

Command Syntax :TIMEbase[:MAIN]:RANGe <range_value>

<range_value> ::= time for 10 div in seconds in NR3 format

The :TIMEbase[:MAIN]: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[:MAIN]:RANGe?

The :TIMEbase[:MAIN]: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 1468
 - [":TIMEbase:MODE"](#) on page 1472
 - [":TIMEbase\[:MAIN\]:SCALE"](#) on page 1471
 - [":TIMEbase:WINDow:RANGe"](#) on page 1477

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 49](#), "Programming Examples," starting on page 1717

:TIMebase[:MAIN]:SCALE

N (see [page 1708](#))

Command Syntax :TIMebase[:MAIN]:SCALE <scale_value>

<scale_value> ::= time/div in seconds in NR3 format

The :TIMebase[:MAIN]:SCALE command sets the horizontal scale or units per division for the main window.

Query Syntax :TIMebase[:MAIN]:SCALE?

The :TIMebase[:MAIN]: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 1468
 - [":TIMebase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TIMebase:WINDow:SCALE"](#) on page 1478
 - [":TIMebase:WINDow:RANGe"](#) on page 1477

:TIMEbase:MODE

C (see [page 1708](#))

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 1468
 - ["*RST \(Reset\)"](#) on page 247
 - [":TIMEbase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TIMEbase\[:MAIN\]:POSition"](#) on page 1469
 - [":TIMEbase:REFerence"](#) on page 1473
 - [":DISPlay:XYColor"](#) on page 451

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 49](#), "Programming Examples," starting on page 1717

:TImebase:REFeRence

C (see [page 1708](#))

Command Syntax :TImebase:REFeRence <reference>

<reference> ::= {LEFT | CENTer | RIGHT | CUSTom}

The :TImebase:REFeRence command sets the time reference to:

- LEFT – one division from the left side of the screen.
- CENTer – the center of the screen.
- RIGHT – one division from the right side of the screen.
- CUSTom – lets you use the :TImebase:REFeRence:LOCation command to place the time reference location at a percent of the graticule width (where 0.0 is the left edge and 1.0 is the right edge).

The 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 | CUST}

- See Also**
- ["Introduction to :TImebase Commands"](#) on page 1468
 - [":TImebase:REFeRence:LOCation"](#) on page 1474
 - [":TImebase:MODE"](#) on page 1472

Example Code myScope.WriteString ":TImebase:REFeRence CENTer" ' Set reference to center.

See complete example programs at: [Chapter 49](#), "Programming Examples," starting on page 1717

:TIMebase:REFeRence:LOCation

N (see [page 1708](#))

Command Syntax :TIMebase:REFeRence:LOCation <loc>

<loc> ::= 0.0 to 1.0 in NR3 format

When the :TIMebase:REFeRence is set to CUSTom, the :TIMebase:REFeRence:LOCation command lets you place the time reference location at a percent of the graticule width (where 0.0 is the left edge and 1.0 is the right edge).

Query Syntax :TIMebase:REFeRence:LOCation?

The :TIMebase:REFeRence:LOCation? query returns the time base reference custom location setting.

Return Format <loc><NL>

<loc> ::= 0.0 to 1.0 in NR3 format

See Also • [":TIMebase:REFeRence"](#) on page 1473

:TIMebase:VERNier

N (see [page 1708](#))

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 1468

:TImebase:WINDow:POSition

C (see [page 1708](#))

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 1468
 - [":TImebase:MODE"](#) on page 1472
 - [":TImebase\[:MAIN\]:POSition"](#) on page 1469
 - [":TImebase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TImebase\[:MAIN\]:SCALE"](#) on page 1471
 - [":TImebase:WINDow:RANGe"](#) on page 1477
 - [":TImebase:WINDow:SCALE"](#) on page 1478

:TIMebase:WINDow:RANGe

C (see [page 1708](#))

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 1468
 - [":TIMebase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TIMebase\[:MAIN\]:POSition"](#) on page 1469
 - [":TIMebase\[:MAIN\]:SCALe"](#) on page 1471

:TImebase:WINDow:SCALe

N (see [page 1708](#))

Command Syntax :TImebase:WINDow:SCALe <scale_value>

<scale_value> ::= scale value in seconds in NR3 format

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.

Query Syntax :TImebase:WINDow:SCALe?

The :TImebase:WINDow:SCALe? query returns the current zoomed window scale setting.

Return Format <scale_value><NL>

<scale_value> ::= current seconds per division for the zoomed window

- See Also**
- ["Introduction to :TImebase Commands"](#) on page 1468
 - [":TImebase\[:MAIN\]:RANGe"](#) on page 1470
 - [":TImebase\[:MAIN\]:POSition"](#) on page 1469
 - [":TImebase\[:MAIN\]:SCALe"](#) on page 1471
 - [":TImebase:WINDow:RANGe"](#) on page 1477

40 :TRIGger Commands

Control the trigger modes and parameters for each trigger type. See:

- **"Introduction to :TRIGger Commands"** on page 1479
- **"General :TRIGger Commands"** on page 1481
- **":TRIGger:DElay Commands"** on page 1497
- **":TRIGger[:EDGE] Commands"** on page 1505
- **":TRIGger:GLITch Commands"** on page 1511 (Pulse Width trigger)
- **":TRIGger:OR Commands"** on page 1520
- **":TRIGger:PATtern Commands"** on page 1522
- **":TRIGger:RUNT Commands"** on page 1530
- **":TRIGger:SHOLd Commands"** on page 1535
- **":TRIGger:TRANSition Commands"** on page 1541
- **":TRIGger:ZONE Commands"** on page 1546

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 1496) 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 1494).

- **Edge triggering** – identifies a trigger by looking for a specified slope and voltage level on a waveform.
- **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.

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 +4.00000000E-08;
:TRIG:EDGE:SOUR CHAN1;LEV +0.0E+00;SLOP POS;REJ OFF;COUP DC;
:TRIG:ZONE1:STAT 0;:TRIG:ZONE2:STAT 0
```

General :TRIGger Commands

Table 167 General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:FORCE (see page 1483)	n/a	n/a
:TRIGger:HFReject {{0 OFF} {1 ON}} (see page 1484)	:TRIGger:HFReject? (see page 1484)	{0 1}
:TRIGger:HOLDoff <holdoff_time> (see page 1485)	:TRIGger:HOLDoff? (see page 1485)	<holdoff_time> ::= 60 ns to 10 s in NR3 format
:TRIGger:HOLDoff:MAXimum <max_holdoff> (see page 1486)	:TRIGger:HOLDoff:MAXimum? (see page 1486)	<max_holdoff> ::= maximum holdoff time in seconds in NR3 format
:TRIGger:HOLDoff:MINimum <min_holdoff> (see page 1487)	:TRIGger:HOLDoff:MINimum? (see page 1487)	<min_holdoff> ::= minimum holdoff time in seconds in NR3 format
:TRIGger:HOLDoff:RANDOM {{0 OFF} {1 ON}} (see page 1488)	:TRIGger:HOLDoff:RANDOM? (see page 1488)	<setting> ::= {0 1}
:TRIGger:HYSTeresis:STATe {{0 OFF} {1 ON}} (see page 1490)	:TRIGger:HYSTeresis:STATe? (see page 1490)	{0 1}
:TRIGger:JFRee {{0 OFF} {1 ON}} (see page 1489)	:TRIGger:JFRee? (see page 1489)	{0 1}
:TRIGger:LEVel:ASeTup (see page 1491)	n/a	n/a
:TRIGger:LEVel:HIGH <level>, <source> (see page 1492)	:TRIGger:LEVel:HIGH? <source> (see page 1492)	<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 1493)	:TRIGger:LEVel:LOW? <source> (see page 1493)	<level> ::= .75 x full-scale voltage from center screen in NR3 format. <source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format

Table 167 General :TRIGger Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:MODE <mode> (see page 1494)	:TRIGger:MODE? (see page 1494)	<mode> ::= {EDGE GLITCh PATtern DELay OR RUNT SHOLd TRANSition {PBUS{1 2} SBUS{1 2}} USB} <return_value> ::= {<mode> <none>} <none> ::= query returns "NONE" if the :TIMEbase:MODE is ROLL or XY
:TRIGger:NREJect {{0 OFF} {1 ON}} (see page 1495)	:TRIGger:NREJect? (see page 1495)	{0 1}
:TRIGger:SWEep <sweep> (see page 1496)	:TRIGger:SWEep? (see page 1496)	<sweep> ::= {AUTO NORMal}

:TRIGger:FORCe

N (see [page 1708](#))

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 1479

:TRIGger:HFReject

C (see [page 1708](#))

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 1479
 - [":TRIGger\[:EDGE\]:REJect"](#) on page 1508

:TRIGger:HOLDoff

C (see [page 1708](#))

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 1479

:TRIGger:HOLDoff:MAXimum

N (see [page 1708](#))

Command Syntax :TRIGger:HOLDoff:MAXimum <max_holdoff>

<max_holdoff> ::= maximum holdoff time in seconds in NR3 format

When the random trigger holdoff mode is enabled (see :TRIGger:HOLDoff:RANDom), the :TRIGger:HOLDoff:MAXimum command specifies the maximum trigger holdoff time.

Query Syntax :TRIGger:HOLDoff:MAXimum?

The :TRIGger:HOLDoff:MAXimum? query returns the maximum random trigger holdoff time setting.

Return Format <max_holdoff><NL>

- See Also**
- [":TRIGger:HOLDoff:MINimum"](#) on page 1487
 - [":TRIGger:HOLDoff:RANDom"](#) on page 1488

:TRIGger:HOLDoff:MINimum

N (see [page 1708](#))

Command Syntax :TRIGger:HOLDoff:MINimum <min_holdoff>

<min_holdoff> ::= minimum holdoff time in seconds in NR3 format

When the random trigger holdoff mode is enabled (see :TRIGger:HOLDoff:RANDom), the :TRIGger:HOLDoff:MINimum command specifies the minimum trigger holdoff time.

Query Syntax :TRIGger:HOLDoff:MINimum?

The :TRIGger:HOLDoff:MINimum? query returns the minimum random trigger holdoff time setting.

Return Format <min_holdoff><NL>

- See Also**
- [":TRIGger:HOLDoff:MAXimum"](#) on page 1486
 - [":TRIGger:HOLDoff:RANDom"](#) on page 1488

:TRIGger:HOLDoff:RANDom

N (see [page 1708](#))

Command Syntax :TRIGger:HOLDoff:RANDom {{0 | OFF} | {1 | ON}}

The :TRIGger:HOLDoff:RANDom command enables or disables the random trigger holdoff mode. This mode randomizes the holdoff time from one acquisition to the next. The randomized holdoff time values will be between the values specified by the :TRIGger:HOLDoff:MINimum and :TRIGger:HOLDoff:MAXimum commands.

The random trigger holdoff mode ensures that the oscilloscope re-arms after each acquisition in a manner that minimizes or eliminates the likelihood of triggering at the beginning of a DDR burst. Randomizing the holdoff time increases the likelihood that the oscilloscope will trigger on different data phases of a multi-phase (8 data transfer) burst. This mode mixes up the traffic pattern the oscilloscope triggers on and is very effective when used on repeating patterns.

Query Syntax :TRIGger:HOLDoff:RANDom?

The :TRIGger:HOLDoff:RANDom? query returns random trigger holdoff mode setting.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":TRIGger:HOLDoff:MAXimum"](#) on page 1486
 - [":TRIGger:HOLDoff:MINimum"](#) on page 1487

:TRIGger:HYSTeresis:STATe

N (see [page 1708](#))

Command Syntax :TRIGger:HYSTeresis:STATe {{0 | OFF} | {1 | ON}}

The :TRIGger:HYSTeresis:STATe command disables or enables the trigger Hysteresis option.

Oscilloscope triggers use hysteresis techniques to reduce false triggering on noisy signals. This oscilloscope's fully digital triggers implement hysteresis as a sensitivity limit. The default hysteresis setting (enabled) delivers optimal triggering performance for almost all signals.

By disabling hysteresis, the sensitivity of the trigger drops to one least significant bit (LSB). It is limited only by the noise floor of the signal and oscilloscope. You must be mindful of signal and measurement noise to avoid false triggers. This mode is often paired with Bandwidth Limit filters or Averaging (see [":CHANnel<n>:BANDwidth"](#) on page 349 or [":ACQuire:TYPE"](#) on page 311).

Query Syntax :TRIGger:HYSTeresis:STATe?

The :TRIGger:HYSTeresis:STATe? query returns the trigger Hysteresis option setting.

Return Format {0 | 1}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":CHANnel<n>:BANDwidth"](#) on page 349
 - [":ACQuire:TYPE"](#) on page 311

:TRIGger:JFRee

N (see [page 1708](#))

Command Syntax :TRIGger:JFRee {{0 | OFF} | {1 | ON}}

The :TRIGger:JFRee command disables or enables the Jitter-Free Trigger option.

Oscilloscope trigger circuitry introduces a certain amount of noise and jitter that result in horizontal error. To correct for this error, in certain trigger modes and at certain timebase ranges, hardware Jitter-Free Trigger correction is used to minimize trigger jitter at the trigger point.

The effect seen on the oscilloscope display is that all plotted acquisitions cross through the same pixel location at the intersection of the trigger level and time=0.

Jitter-Free Trigger correction works well for fast digital edges, but you may want to turn it off for some slow slew rate analog signals.

Query Syntax :TRIGger:JFRee?

The :TRIGger:JFRee? query returns the Jitter-Free Trigger option setting.

Return Format {0 | 1}

:TRIGger:LEVel:ASETup

N (see [page 1708](#))

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 1507

:TRIGger:LEVel:HIGH

N (see [page 1708](#))

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 1479
 - [":TRIGger:LEVel:LOW"](#) on page 1493
 - [":TRIGger:RUNT Commands"](#) on page 1530
 - [":TRIGger:TRANSition Commands"](#) on page 1541
 - [":TRIGger\[:EDGE\]:SOURce"](#) on page 1510

:TRIGger:LEVel:LOW

N (see [page 1708](#))

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 1479
 - [":TRIGger:LEVel:HIGH"](#) on page 1492
 - [":TRIGger:RUNT Commands"](#) on page 1530
 - [":TRIGger:TRANSition Commands"](#) on page 1541
 - [":TRIGger\[:EDGE\]:SOURce"](#) on page 1510

:TRIGger:MODE

C (see [page 1708](#))

Command Syntax :TRIGger:MODE <mode>

```
<mode> ::= {EDGE | GLITCh | PATTeRn | DELay | OR | RUNT | SHOLd
            | TRANsition | {PBUS{1 | 2} | SBUS{1 | 2}}}
```

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 | DEL | OR | RUNT | SHOL
            | TRAN | PBUS{1 | 2}}
```

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:SWEep"](#) on page 1496
 - [":TIMebase:MODE"](#) on page 1472

Example Code

```
' TRIGGER_MODE - Set the trigger mode to EDGE.
myScope.WriteString ":TRIGger:MODE EDGE"
```

See complete example programs at: [Chapter 49](#), "Programming Examples," starting on page 1717

:TRIGger:NREJect

C (see [page 1708](#))

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.

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 1479

:TRIGger:SWEep

C (see [page 1708](#))

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 1479

:TRIGger:DElay Commands

Table 168 :TRIGger:DElay Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DElay:ARM:SL OPe <slope> (see page 1499)	:TRIGger:DElay:ARM:SL OPe? (see page 1499)	<slope> ::= {NEGative POSitive}
:TRIGger:DElay:ARM:SO URce <source> (see page 1500)	:TRIGger:DElay:ARM:SO URce? (see page 1500)	<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 :TMAX <time_value> (see page 1501)	:TRIGger:DElay:TDElay :TMAX? (see page 1501)	<time_value> ::= time in seconds in NR3 format
:TRIGger:DElay:TDElay :TMIN <time_value> (see page 1502)	:TRIGger:DElay:TDElay :TMIN? (see page 1502)	<time_value> ::= time in seconds in NR3 format
:TRIGger:DElay:TRIGge r:SLOPe <slope> (see page 1503)	:TRIGger:DElay:TRIGge r:SLOPe? (see page 1503)	<slope> ::= {NEGative POSitive}
:TRIGger:DElay:TRIGge r:SOURce <source> (see page 1504)	:TRIGger:DElay:TRIGge r:SOURce? (see page 1504)	<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.

The Edge Then Edge trigger mode triggers when the trigger edge occurs after an arming edge within the minimum and maximum delay times without any intervening arming or triggering edges. The sequence is restarted without a trigger if:

- An arming or triggering edge occurs while waiting for the minimum delay time to complete.
- An arming edge occurs before a triggering edge during the maximum delay time.
- The maximum delay time completes with no triggering edge.

NOTE

This Edge Then Edge trigger mode differs from previous InfiniiVision X-Series oscilloscopes where intervening edges were allowed (you could trigger on an Nth triggering edge) and there was no maximum delay time.

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 1708](#))

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 1479
 - [":TRIGger:DElay:ARM:SOURce"](#) on page 1500
 - [":TRIGger:DElay:TDElay:TMIN"](#) on page 1502
 - [":TRIGger:DElay:TDElay:TMAX"](#) on page 1501

:TRIGger:DElay:ARM:SOURce

N (see [page 1708](#))

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 1479
 - [":TRIGger:DElay:ARM:SLOPe"](#) on page 1499
 - [":TRIGger:DElay:TDElay:TMIN"](#) on page 1502
 - [":TRIGger:DElay:TDElay:TMAX"](#) on page 1501
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:DElay:TDElay:TMAX

N (see [page 1708](#))

Command Syntax :TRIGger:DElay:TDElay:TMAX <time_value>

<time_value> ::= time in seconds in NR3 format

The :TRIGger:DElay:TDElay:TMAX command sets the maximum delay time between the arming edge and the trigger edge in the Edge Then Edge trigger.

NOTE

Entered delay times automatically adjust to the nearest valid increment of ~156.5 ps.

Query Syntax :TRIGger:DElay:TDElay:TMAX?

The :TRIGger:DElay:TDElay:TMAX? query returns current maximum delay time setting.

Return Format <time_value><NL>

<time_value> ::= time in seconds in NR3 format

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:DElay:TDElay:TMIN"](#) on page 1502
 - [":TRIGger:DElay:TRIGger:SLOPe"](#) on page 1503

:TRIGger:DElay:TDElay:TMIN

N (see [page 1708](#))

Command Syntax :TRIGger:DElay:TDElay:TMIN <time_value>

<time_value> ::= time in seconds in NR3 format

The :TRIGger:DElay:TDElay:TMIN command sets the minimum delay time between the arming edge and the trigger edge in the Edge Then Edge trigger.

NOTE

Entered delay times automatically adjust to the nearest valid increment of ~156.5 ps.

Query Syntax :TRIGger:DElay:TDElay:TMIN?

The :TRIGger:DElay:TDElay:TMIN? query returns current minimum delay time setting.

Return Format <time_value><NL>

<time_value> ::= time in seconds in NR3 format

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:DElay:TDElay:TMAX"](#) on page 1501
 - [":TRIGger:DElay:TRIGger:SLOPe"](#) on page 1503

:TRIGger:DElay:TRIGger:SLOPe

N (see [page 1708](#))

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 1479
 - [":TRIGger:DElay:TRIGger:SOURce"](#) on page 1504
 - [":TRIGger:DElay:TDElay:TMIN"](#) on page 1502
 - [":TRIGger:DElay:TDElay:TMAX"](#) on page 1501

:TRIGger:DElay:TRIGger:SOURce

N (see [page 1708](#))

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 1479
 - [":TRIGger:DElay:TRIGger:SLOPe"](#) on page 1503
 - [":TRIGger:DElay:TDElay:TMIN"](#) on page 1502
 - [":TRIGger:DElay:TDElay:TMAX"](#) on page 1501
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger[:EDGE] Commands

Table 169 :TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:COUPling {AC DC LFReject} (see page 1506)	:TRIGger[:EDGE]:COUPling? (see page 1506)	{AC DC LFReject}
:TRIGger[:EDGE]:LEVel <level> [, <source>] (see page 1507)	:TRIGger[:EDGE]:LEVel? [<source>] (see page 1507)	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, <level> ::= ±8 V. <source> ::= {CHANnel<n> DIGital<d> EXTernal } <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 1508)	:TRIGger[:EDGE]:REJect? (see page 1508)	{OFF LFReject HFReject}
:TRIGger[:EDGE]:SLOPe <polarity> (see page 1509)	:TRIGger[:EDGE]:SLOPe? (see page 1509)	<polarity> ::= {POSitive NEGative EITHer ALTernate}
:TRIGger[:EDGE]:SOURC e <source> (see page 1510)	:TRIGger[:EDGE]:SOURC e? (see page 1510)	<source> ::= {CHANnel<n> DIGital<d> EXTernal LINE WGEN WGEN1 WMOD} <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

C (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger\[:EDGE\]:REJect"](#) on page 1508

:TRIGger[:EDGE]:LEVel

C (see [page 1708](#))

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> ::= $\pm$ (external range setting) in NR3 format for external triggers

<level> ::= ± 8 V for digital channels

<source> ::= {CHANnel<n> | DIGital<d> | EXTernal}

<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 1479
 - [":TRIGger\[:EDGE\]:SOURce"](#) on page 1510
 - [":EXTernal:RANGe"](#) on page 462
 - [":POD<n>:THReshold"](#) on page 835
 - [":DIGital<d>:THReshold"](#) on page 404

:TRIGger[:EDGE]:REJect

C (see [page 1708](#))

Command Syntax :TRIGger[:EDGE]:REJect <reject>
 <reject> ::= {OFF | LFReject | 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 1479
 - [":TRIGger:HFReject"](#) on page 1484
 - [":TRIGger\[:EDGE\]:COUpling"](#) on page 1506

:TRIGger[:EDGE]:SLOPe

C (see [page 1708](#))

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.

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 1479
 - [":TRIGger:MODE"](#) on page 1494

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 49](#), "Programming Examples," starting on page 1717

:TRIGger[:EDGE]:SOURce

C (see [page 1708](#))

Command Syntax :TRIGger[:EDGE]:SOURce <source>

<source> ::= {CHANnel<n> | DIGital<d> | EXTernal | LINE | WGEN | WGEN1
| WMOD}

<n> ::= 1 to (# analog channels) in NR1 format

<d> ::= 0 to (# digital channels - 1) in NR1 format

Note: 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 FM modulation is used, triggers at the 50% level of the rising edge of the modulating signal.

Query Syntax :TRIGger[:EDGE]:SOURce?

The :TRIGger[:EDGE]:SOURce? query returns the current source. If all channels are off, the query returns "NONE."

Return Format <source><NL>

<source> ::= {CHAN<n> | DIG<d> | EXTernal | LINE | WGEN | WGEN1
| WMOD | NONE}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494

Example Code

```
' TRIGGER_EDGE_SOURCE - Selects the channel that actually produces the
edge trigger. Any channel can be selected.
myScope.WriteString ":TRIGger:EDGE:SOURce CHANnel1"
```

See complete example programs at: [Chapter 49](#), "Programming Examples," starting on page 1717

:TRIGger:GLITCh Commands

Table 170 :TRIGger:GLITCh Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITCh:GREAt erthan <greater_than_time>[s uffix] (see page 1513)	:TRIGger:GLITCh:GREAt erthan? (see page 1513)	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:GLITCh:LESt han <less_than_time>[suff ix] (see page 1514)	:TRIGger:GLITCh:LESt han? (see page 1514)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:GLITCh:LEVel <level> [<source>] (see page 1515)	:TRIGger:GLITCh:LEVel ? (see page 1515)	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For digital channels, <level> ::= ± 8 V. <source> ::= {CHANnel<n> DIGital<d>} <n> ::= 1 to (# analog channels) in NR1 format <d> ::= 0 to (# digital channels - 1) in NR1 format
:TRIGger:GLITCh:POLar ity <polarity> (see page 1516)	:TRIGger:GLITCh:POLar ity? (see page 1516)	<polarity> ::= {POSitive NEGative}
:TRIGger:GLITCh:QUALi fier <qualifier> (see page 1517)	:TRIGger:GLITCh:QUALi fier? (see page 1517)	<qualifier> ::= {GREaterthan LESSthan RANGE}

Table 170 :TRIGger:GLITCh Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:GLITCh:RANGe <less_than_time>[suffix], <greater_than_time>[suffix] (see page 1518)	:TRIGger:GLITCh:RANGe? (see page 1518)	<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 1519)	:TRIGger:GLITCh:SOURCe? (see page 1519)	<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 1708](#))

- 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 1479
 - [":TRIGger:GLITch:SOURce"](#) on page 1519
 - [":TRIGger:GLITch:QUALifier"](#) on page 1517
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:GLITch:LESSthan

N (see [page 1708](#))

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 1479
 - [":TRIGger:GLITch:SOURce"](#) on page 1519
 - [":TRIGger:GLITch:QUALifier"](#) on page 1517
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:GLITch:LEVel

N (see [page 1708](#))

Command Syntax :TRIGger:GLITch:LEVel <level>[,<source>]

<level> ::= .75 x full-scale voltage from center screen in NR3 format
for internal triggers

<level> ::= ± 8 V for digital channels

<source> ::= {CHANnel<n> | DIGital<d>}

<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><NL>

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:GLITch:SOURce"](#) on page 1519
 - [":EXTernal:RANGe"](#) on page 462

:TRIGger:GLITch:POLarity

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:GLITch:SOURce"](#) on page 1519

:TRIGger:GLITch:QUALifier

N (see [page 1708](#))

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 1479
 - [":TRIGger:GLITch:SOURce"](#) on page 1519
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:GLITch:RANGe

N (see [page 1708](#))

```

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

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

- [":TRIGger:GLITCh:SOURce"](#) on page 1479
- [":TRIGger:GLITCh:QUALifier"](#) on page 1519
- [":TRIGger:GLITCh:QUALifier"](#) on page 1517
- [":TRIGger:MODE"](#) on page 1494

:TRIGger:GLITch:SOURce

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:GLITch:LEVel"](#) on page 1515
 - [":TRIGger:GLITch:POLarity"](#) on page 1516
 - [":TRIGger:GLITch:QUALifier"](#) on page 1517
 - [":TRIGger:GLITch:RANGe"](#) on page 1518

Example Code • ["Example Code"](#) on page 1510

:TRIGger:OR Commands

Table 171 :TRIGger:OR Commands Summary

Command	Query	Options and Query Returns
:TRIGger:OR <string> (see page 1521)	:TRIGger:OR? (see page 1521)	<string> ::= "nn...n" where n ::= {R F E X} R = rising edge, F = falling edge, E = either edge, X = don't care. Each character in the string is for an analog or digital channel as shown on the front panel display.

:TRIGger:OR

N (see [page 1708](#))

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
4 analog + 16 digital channels (mixed-signal)	Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 4 through 1.
2 analog + 16 digital channels (mixed-signal)	Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - 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 1479
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:PATtern Commands

Table 172 :TRIGger:PATtern Commands Summary

Command	Query	Options and Query Returns
:TRIGger:PATtern <string>[,<edge_source>,<edge>] (see page 1523)	:TRIGger:PATtern? (see page 1524)	<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> DIGital<d> NONE} <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 1525)	:TRIGger:PATtern:FORM at? (see page 1525)	<base> ::= {ASCii HEX}
:TRIGger:PATtern:GREa terthan <greater_than_time>[suffix] (see page 1526)	:TRIGger:PATtern:GREa terthan? (see page 1526)	<greater_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:PATtern:LESS than <less_than_time>[suffix] (see page 1527)	:TRIGger:PATtern:LESS than? (see page 1527)	<less_than_time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:PATtern:QUAL ifier <qualifier> (see page 1528)	:TRIGger:PATtern:QUAL ifier? (see page 1528)	<qualifier> ::= {ENTered GREaterthan LESSthan INRange OUTRange TIMEout}
:TRIGger:PATtern:RANG e <less_than_time>[suffix], <greater_than_time>[suffix] (see page 1529)	:TRIGger:PATtern:RANG e? (see page 1529)	<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 1708](#))

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> | DIGital<d>
| NONE}

<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
4 analog + 16 digital channels (mixed-signal)	Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 4 through 1.
2 analog + 16 digital channels (mixed-signal)	Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - 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 1479
 - [":TRIGger:PATtern:FORMat"](#) on page 1525
 - [":TRIGger:PATtern:QUALifier"](#) on page 1528
 - [":TRIGger:MODE"](#) on page 1494
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

:TRIGger:PATtern:FORMat

N (see [page 1708](#))

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 1479
 - [":TRIGger:PATtern"](#) on page 1523

:TRIGger:PATtern:GREaterthan

N (see [page 1708](#))

```

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

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 1479
- [":TRIGger:PATtern"](#) on page 1523
- [":TRIGger:PATtern:QUALifier"](#) on page 1528
- [":TRIGger:MODE"](#) on page 1494

:TRIGger:PATtern:LESSthan

N (see [page 1708](#))

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 1479
 - [":TRIGger:PATtern"](#) on page 1523
 - [":TRIGger:PATtern:QUALifier"](#) on page 1528
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:PATtern:QUALifier

N (see [page 1708](#))

Command Syntax :TRIGger:PATtern:QUALifier <qualifier>

<qualifier> ::= {ENTERed | GREaterthan | LESSthan | INRange | OUTRange
| TIMEout}

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.

Query Syntax :TRIGger:PATtern:QUALifier?

The :TRIGger:PATtern:QUALifier? query returns the trigger duration qualifier.

Return Format <qualifier><NL>

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:PATtern:GREaterthan"](#) on page 1526
 - [":TRIGger:PATtern:LESSthan"](#) on page 1527
 - [":TRIGger:PATtern:RANGE"](#) on page 1529

:TRIGger:PATTern:RANGe

N (see [page 1708](#))

```

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 1479
- **":TRIGger:PATtern"** on page 1523
- **":TRIGger:PATtern:QUALifier"** on page 1528
- **":TRIGger:MODE"** on page 1494

:TRIGger:RUNT Commands

Table 173 :TRIGger:RUNT Commands Summary

Command	Query	Options and Query Returns
:TRIGger:RUNT:POLarity <polarity> (see page 1531)	:TRIGger:RUNT:POLarity? (see page 1531)	<polarity> ::= {POSitive NEGative}
:TRIGger:RUNT:QUALifier <qualifier> (see page 1532)	:TRIGger:RUNT:QUALifier? (see page 1532)	<qualifier> ::= {GREaterthan LESSthan NONE}
:TRIGger:RUNT:SOURce <source> (see page 1533)	:TRIGger:RUNT:SOURce? (see page 1533)	<source> ::= CHANnel<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:RUNT:TIME <time>[suffix] (see page 1534)	:TRIGger:RUNT:TIME? (see page 1534)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

:TRIGger:RUNT:POLarity

N (see [page 1708](#))

Command Syntax :TRIGger:RUNT:POLarity <polarity>
 <polarity> ::= {POSitive | NEGative}

The :TRIGger:RUNT:POLarity command sets the polarity for the runt trigger:

- POSitive – positive runt pulses.
- NEGative – 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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:LEVel:HIGH"](#) on page 1492
 - [":TRIGger:LEVel:LOW"](#) on page 1493
 - [":TRIGger:RUNT:SOURce"](#) on page 1533

:TRIGger:RUNT:QUALifier

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:RUNT:TIME"](#) on page 1534

:TRIGger:RUNT:SOURce

N (see [page 1708](#))

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 1479
 - [":TRIGger:RUNT:POLarity"](#) on page 1531

:TRIGger:RUNT:TIME

N (see [page 1708](#))

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 1479
 - [":TRIGger:RUNT:QUALifier"](#) on page 1532

:TRIGger:SHOLd Commands

Table 174 :TRIGger:SHOLd Commands Summary

Command	Query	Options and Query Returns
:TRIGger:SHOLd:SLOPe <slope> (see page 1536)	:TRIGger:SHOLd:SLOPe? (see page 1536)	<slope> ::= {NEGative POSitive}
:TRIGger:SHOLd:SOURce :CLOCK <source> (see page 1537)	:TRIGger:SHOLd:SOURce :CLOCK? (see page 1537)	<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 1538)	:TRIGger:SHOLd:SOURce :DATA? (see page 1538)	<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 1539)	:TRIGger:SHOLd:TIME:H OLD? (see page 1539)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}
:TRIGger:SHOLd:TIME:S ETup <time>[suffix] (see page 1540)	:TRIGger:SHOLd:TIME:S ETup? (see page 1540)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

:TRIGger:SHOLd:SLOPe

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:SHOLd:SOURce:CLOCK"](#) on page 1537
 - [":TRIGger:SHOLd:SOURce:DATA"](#) on page 1538

:TRIGger:SHOLd:SOURce:CLOCK

N (see [page 1708](#))

Command Syntax :TRIGger:SHOLd:SOURce:CLOCK <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:SHOLd:SOURce:CLOCK command selects the input channel probing the clock signal.

Query Syntax :TRIGger:SHOLd:SOURce:CLOCK?

The :TRIGger:SHOLd:SOURce:CLOCK? query returns the currently set clock signal source.

Return Format <source><NL>

<source> ::= {CHAN<n> | DIG<d>}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:SHOLd:SLOPe"](#) on page 1536

:TRIGger:SHOLd:SOURce:DATA

N (see [page 1708](#))

Command Syntax :TRIGger:SHOLd:SOURce:DATA <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:SHOLd:SOURce:DATA command selects the input channel probing the data signal.

Query Syntax :TRIGger:SHOLd:SOURce:DATA?

The :TRIGger:SHOLd:SOURce:DATA? query returns the currently set data signal source.

Return Format <source><NL>

<source> ::= {CHAN<n> | DIG<d>}

- See Also**
- ["Introduction to :TRIGger Commands"](#) on page 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:SHOLd:SLOPe"](#) on page 1536

:TRIGger:SHOLd:TIME:HOLD

N (see [page 1708](#))

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 1479

:TRIGger:SHOLd:TIME:SETup

N (see [page 1708](#))

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 1479

:TRIGger:TRANSition Commands

The :TRIGger:TRANSition commands set the rise/fall time trigger options.

Table 175 :TRIGger:TRANSition Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TRANSition:QUALifier <qualifier> (see page 1542)	:TRIGger:TRANSition:QUALifier? (see page 1542)	<qualifier> ::= {GREATERthan LESSthan}
:TRIGger:TRANSition:SLOPe <slope> (see page 1543)	:TRIGger:TRANSition:SLOPe? (see page 1543)	<slope> ::= {NEGative POSitive}
:TRIGger:TRANSition:SOURce <source> (see page 1544)	:TRIGger:TRANSition:SOURce? (see page 1544)	<source> ::= CHANNEL<n> <n> ::= 1 to (# analog channels) in NR1 format
:TRIGger:TRANSition:TIME <time>[suffix] (see page 1545)	:TRIGger:TRANSition:TIME? (see page 1545)	<time> ::= floating-point number in NR3 format [suffix] ::= {s ms us ns ps}

:TRIGger:TRANSition:QUALifier

N (see [page 1708](#))

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 1479
 - [":TRIGger:TRANSition:TIME"](#) on page 1545
 - [":TRIGger:MODE"](#) on page 1494

:TRIGger:TRANSition:SLOPe

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:TRANSition:SOURce"](#) on page 1544

:TRIGger:TRANSition:SOURce

N (see [page 1708](#))

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 1479
 - [":TRIGger:MODE"](#) on page 1494
 - [":TRIGger:TRANSition:SLOPe"](#) on page 1543

:TRIGger:TRANSition:TIME

N (see [page 1708](#))

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 1479
 - [":TRIGger:TRANSition:QUALifier"](#) on page 1542

:TRIGger:ZONE Commands

Table 176 :TRIGger:ZONE Commands Summary

Command	Query	Options and Query Returns
:TRIGger:ZONE<z>:LOGic {AND OR} (see page 1547)	:TRIGger:ZONE<z>:LOGic? (see page 1551)	{AND OR} <z> ::= 2-4 in NR1 format
:TRIGger:ZONE<z>:MODE <mode> (see page 1548)	:TRIGger:ZONE<z>:MODE ? (see page 1548)	<mode> ::= {INTERsect NOTintersect} <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:PLACement <width>, <height>, <x_center>, <y_center> (see page 1549)	:TRIGger:ZONE<z>:PLACement? (see page 1549)	<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 <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:SOURce <source> (see page 1550)	:TRIGger:ZONE<z>:SOURce? (see page 1550)	<source> ::= {CHANnel<n>} <n> ::= 1 to (# analog channels) in NR1 format <z> ::= 1-4 in NR1 format
:TRIGger:ZONE<z>:STATe {{0 OFF} {1 ON}} (see page 1551)	:TRIGger:ZONE<z>:STATe? (see page 1551)	{0 1} <z> ::= 1-4 in NR1 format
n/a	:TRIGger:ZONE<z>:VALidity? (see page 1552)	<value> ::= {VALid INValid OSCreen} <z> ::= 1-4 in NR1 format

:TRIGger:ZONE<z>:LOGic

N (see [page 1708](#))

Command Syntax :TRIGger:ZONE<z>:LOGic {AND | OR}

<z> ::= 2-4 in NR1 format

For Zones 2, 3, or 4, the :TRIGger:ZONE<z>:LOGic command sets the logic to be used when combining with the previous Zone.

The overall Zone Logic is displayed in the grid and in the Zone Trigger dialog box. AND combinations occur before OR combinations.

Query Syntax :TRIGger:ZONE<n>:LOGic?

The :TRIGger:ZONE<n>:LOGic? query returns the logic to be used when combining with the previous Zone.

Return Format <logic><NL>

<logic> ::= {AND | OR}

See Also • [":TRIGger:ZONE<z>:STATe"](#) on page 1551

:TRIGger:ZONE<z>:MODE

N (see [page 1708](#))

Command Syntax :TRIGger:ZONE<z>:MODE <mode>

<mode> ::= {INTersect | NOTintersect}

<z> ::= 1-4 in NR1 format

The :TRIGger:ZONE<z>:MODE command sets the zone qualifying condition for the Zone as either "Must Intersect" or "Must Not Intersect".

Query Syntax :TRIGger:ZONE<z>:MODE?

The :TRIGger:ZONE<z>:MODE? query returns the zone qualifying condition for the Zone.

Return Format <mode><NL>

<mode> ::= {INT | NOT}

- See Also**
- [":TRIGger:ZONE<z>:STATe"](#) on page 1551
 - [":TRIGger:ZONE<z>:PLACement"](#) on page 1549
 - [":TRIGger:ZONE<z>:VALidity?"](#) on page 1552

:TRIGger:ZONE<z>:PLACement

N (see [page 1708](#))

Command Syntax :TRIGger:ZONE<z>: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

<z> ::= 1-4 in NR1 format

The :TRIGger:ZONE<z>:PLACement command sets the size and location of a Zone.

No error is returned if the zone is placed off-screen, or if the zones overlap such that the Zone becomes invalid. The :TRIGger:ZONE<z>:VALidity? query is used to retrieve this information.

Query Syntax :TRIGger:ZONE<z>:PLACement?

The :TRIGger:ZONE<z>:PLACement? query returns the size and location of the Zone.

Return Format <opt><NL>

<opt> ::= <width>, <height>, <x_center>, <y_center>

- See Also**
- [":TRIGger:ZONE<z>:STATe"](#) on page 1551
 - [":TRIGger:ZONE<z>:MODE"](#) on page 1548
 - [":TRIGger:ZONE<z>:VALidity?"](#) on page 1552

:TRIGger:ZONE<z>:SOURce

N (see [page 1708](#))

Command Syntax :TRIGger:ZONE<z>:SOURce <source>

<source> ::= {CHANnel<n>}

<n> ::= 1 to (# analog channels) in NR1 format

<z> ::= 1-4 in NR1 format

The :TRIGger:ZONE<z>:SOURce command sets the analog source channel for a particular Zone.

Query Syntax :TRIGger:ZONE<z>:SOURce?

The :TRIGger:ZONE<z>:SOURce? query returns the analog source channel specified for a particular Zone.

Return Format <source><NL>

<source> ::= {CHAN<n>}

- See Also**
- [":TRIGger:ZONE<z>:MODE"](#) on page 1548
 - [":TRIGger:ZONE<z>:PLACement"](#) on page 1549
 - [":TRIGger:ZONE<z>:STATe"](#) on page 1551
 - [":TRIGger:ZONE<z>:VALidity?"](#) on page 1552

:TRIGger:ZONE<z>:STATe

N (see [page 1708](#))

Command Syntax :TRIGger:ZONE<z>:STATe <on_off>

<on_off> ::= {{0 | OFF} | {1 | ON}}

<z> ::= 1-4 in NR1 format

The :TRIGger:ZONE<n>:STATe command specifies whether a Zone is ON or OFF.

When a zone's state is on, that zone is actively being used to qualify the trigger if it is not invalid or off-screen (see [":TRIGger:ZONE<z>:VALidity?"](#) on page 1552).

Query Syntax :TRIGger:ZONE<z>:STATe?

The :TRIGger:ZONE<z>:STATe? query returns whether a Zone is ON or OFF.

Return Format <on_off><NL>

<on_off> ::= {0 | 1}

See Also • [":TRIGger:ZONE<z>:VALidity?"](#) on page 1552

:TRIGger:ZONE<z>:VALidity?

N (see [page 1708](#))

Query Syntax :TRIGger:ZONE<z>:VALidity?

<z> ::= 1-4 in NR1 format

The :TRIGger:ZONE<z>:VALidity? query returns the validity of the specified Zone.

- INValid is returned when a Zone overlaps and has 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<z>:STATe"](#) on page 1551
 - [":TRIGger:ZONE<z>:MODE"](#) on page 1548
 - [":TRIGger:ZONE<z>:PLACement"](#) on page 1549

41 :WAVeform Commands

Provide access to waveform data. See ["Introduction to :WAVeform Commands"](#) on page 1555.

Table 177 :WAVeform Commands Summary

Command	Query	Options and Query Returns
:WAVeform:BYTeorder <value> (see page 1561)	:WAVeform:BYTeorder? (see page 1561)	<value> ::= {LSBFirst MSBFirst}
n/a	:WAVeform:COUNt? (see page 1562)	<count> ::= an integer from 1 to 65536 in NR1 format
n/a	:WAVeform:DATA? (see page 1563)	<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 1565)	:WAVeform:FORMat? (see page 1565)	<value> ::= {WORD BYTE ASCII}
:WAVeform:POINTs <wfm_points> (see page 1566)	:WAVeform:POINTs? (see page 1566)	<wfm_points> ::= <#_points> <#_points> ::= an integer in NR1 format
:WAVeform:POINTs:MODE <points_mode> (see page 1568)	:WAVeform:POINTs:MODE ? (see page 1569)	<points_mode> ::= {NORMal MAXimum RAW}

Table 177 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVeform:PREAmble? (see page 1570)	<pre><preamble_block> ::= <format NR1>, <type NR1>,<points NR1>,<count NR1>, <xincrement NR3>, <xorigin NR3>, <xreference NR1>,<yincrement NR3>, <yorigin NR3>, <yreference NR1></pre> <pre><format> ::= an integer in NR1 format:</pre> • 0 for BYTE format • 1 for WORD format • 2 for ASCII format <pre><type> ::= an integer in NR1 format:</pre> • 0 for NORMAl type • 1 for PEAK detect type • 3 for AVERAge type <pre><count> ::= Average count, or 1 if PEAK detect type or NORMAl; an integer in NR1 format</pre>
:WAVeform:SEGmented:ALL {{0 OFF} {1 ON}} (see page 1573)	:WAVeform:SEGmented:ALL? (see page 1573)	<pre><setting> ::= {0 1}</pre>
n/a	:WAVeform:SEGmented:COUNT? (see page 1574)	<pre><count> ::= an integer from 2 to 1000 in NR1 format</pre>
n/a	:WAVeform:SEGmented:TAG? (see page 1575)	<pre><time_tag> ::= in NR3 format</pre>
n/a	:WAVeform:SEGmented:XLIST? <xlist_type> (see page 1576)	<pre><xlist_type> ::= {RELXorigin ABSXorigin TTAG}</pre> <pre><return_value> ::= X-info for all segments</pre>
:WAVeform:SOURce <source> (see page 1577)	:WAVeform:SOURce? (see page 1577)	<pre><source> ::= {CHANnel<n> POD{1 2} BUS{1 2} FUNCTION<m> MATH<m> FFT {PBUS{1 2} SBUS{1 2}}}</pre> <pre><n> ::= 1 to (# analog channels) in NR1 format</pre> <pre><m> ::= 1 to (# math functions) in NR1 format</pre>
:WAVeform:SOURce:SUBSource <subsource> (see page 1581)	:WAVeform:SOURce:SUBSource? (see page 1581)	<pre><subsource> ::= {{SUB0 RX MOSI FAST} {SUB1 TX MISO SLOW}}</pre>

Table 177 :WAVEform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVEform:TYPE? (see page 1582)	<return_mode> ::= {NORM PEAK AVER}
:WAVEform:UNSigned { {0 OFF} {1 ON} } (see page 1583)	:WAVEform:UNSigned? (see page 1583)	{0 1}
:WAVEform:VIEW <view> (see page 1584)	:WAVEform:VIEW? (see page 1584)	<view> ::= {MAIN ALL}
n/a	:WAVEform:XINCrement? (see page 1585)	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVEform:XORigin? (see page 1586)	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVEform:XREFerence? (see page 1587)	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)
n/a	:WAVEform:YINCrement? (see page 1588)	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVEform:YORigin? (see page 1589)	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVEform:YREFerence? (see page 1590)	<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 1563](#)) and :WAVEform:PREAmble (see [page 1570](#)). 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 311](#)): NORMal, AVERage, and PEAK. Digital channels are always acquired using NORMal. When the data is acquired using the :DIGitize command (see [page 271](#)) or :RUN command (see [page 275](#)), 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 1563](#)) 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[:ANALog]? (see [page 297](#)).

Helpful Hints:

The number of points transferred to the computer is controlled using the :WAVEform:POINTS command (see [page 1566](#)). 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 1566](#)). 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 288](#)). 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 1566](#)). 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 1566](#)). 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 311](#)), the value returned by the :WAVEform:XINCrement query (see [page 1585](#)) should be doubled to find the time difference between the min-max pairs.

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:

$$\text{voltage} = [(\text{data value} - \text{yreference}) * \text{yincrement}] + \text{yorigin}$$

If the :WAVEform:FORMat data format is ASCII (see [page 1565](#)), 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:

$$\text{time} = [(\text{data point number} - \text{xreference}) * \text{xincrement}] + \text{xorigin}$$

This would result in the following calculation for time bucket 3:

$$\text{time} = [(3 - 0) * 2 \text{ ns}] + 16 \text{ ns} = 22 \text{ ns}$$

In :ACQuire:TYPE PEAK mode (see [page 311](#)), because data is acquired in max-min pairs, modify the previous time formula to the following:

$$\text{time} = [(\text{data pair number} - \text{xreference}) * \text{xincrement} * 2] + \text{xorigin}$$

Data Format for Transfer

There are three formats for transferring waveform data over the interface: BYTE, WORD and ASCII (see [":WAVEform:FORMat"](#) on page 1565). 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 1583](#)) 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 1565) 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 1561](#)) and :WAVEform:UNSIGNED (see [page 1583](#)) have no effect when the format is ASCII.

Data Format for Transfer - WORD format

WORD format (see [":WAVEform:FORMat"](#) on page 1565) 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 1566](#)). 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 1561](#)) 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 1565](#)) 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 1561](#)) has no effect when the data format is BYTE.

Digital Channel Data

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 1583](#)) 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 1565](#)) is WORD, every other data byte will be 0. The setting of :WAVeform:BYTeorder (see [page 1561](#)) 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 313](#)) 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 +32000;SOUR CHAN1;  
SOUR:SUBS SUB0
```

:WAVEform:BYTeorder

C (see [page 1708](#))

Command Syntax :WAVEform:BYTeorder <value>

<value> ::= {LSBFirst | MSBFirst}

The :WAVEform:BYTeorder command sets the output sequence of the WORD data.

- MSBFirst – sets the most significant byte to be transmitted first.
- 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 MSBFirst.

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 1555
 - [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:FORMat"](#) on page 1565
 - [":WAVEform:PREamble?"](#) on page 1570

- Example Code**
- ["Example Code"](#) on page 1578
 - ["Example Code"](#) on page 1571

:WAVEform:COUNT?

C (see [page 1708](#))

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 1555
 - [":ACQuire:COUNT"](#) on page 288
 - [":ACQuire:TYPE"](#) on page 311

:WAVEform:DATA?

C (see [page 1708](#))

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 0x0100 – Clipped low. These are locations where the waveform is clipped at the bottom of the oscilloscope display.
- 0xFF or 0xFF00 – 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 1555
 - "[:WAVEform:UNSIGNED](#)" on page 1583
 - "[:WAVEform:BYTEORDER](#)" on page 1561
 - "[:WAVEform:FORMAT](#)" on page 1565
 - "[:WAVEform:POINTS](#)" on page 1566
 - "[:WAVEform:PREAmble?](#)" on page 1570
 - "[:WAVEform:SOURce](#)" on page 1577
 - "[:WAVEform:TYPE?](#)" on page 1582

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) - 1
    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 49](#), "Programming Examples," starting on page 1717

:WAVEform:FORMat

C (see [page 1708](#))

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 1555
 - [":WAVEform:BYTeorder"](#) on page 1561
 - [":WAVEform:SOURce"](#) on page 1577
 - [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:PREamble?"](#) on page 1570

Example Code • ["Example Code"](#) on page 1578

:WAVEform:POINts

C (see [page 1708](#))

Command Syntax :WAVEform:POINts <#_points>
 <#_points> ::= an integer in NR1 format

The :WAVEform:POINts command sets the desired 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 1568](#)) for more information.

To get the maximum number of points possible, you can set the waveform points mode to MAXimum and ask for the maximum memory depth of the oscilloscope.

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 1568](#)) for more information).

In any waveform points mode, the oscilloscope returns as close to the requested number of points as possible, but the actual number of points available depends on oscilloscope settings.

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> ::= an integer number of waveform points

- See Also**
- ["Introduction to :WAVEform Commands"](#) on page 1555
 - [":ACQuire:POINts\[:ANALog\]"](#) on page 297
 - [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:SOURce"](#) on page 1577
 - [":WAVEform:VIEW"](#) on page 1584
 - [":WAVEform:PREamble?"](#) on page 1570
 - [":WAVEform:POINts:MODE"](#) on page 1568

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 49**, "Programming Examples," starting on page 1717

:WAVEform:POINts:MODE

N (see [page 1708](#))

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 *analysis record* and defaults to a 64K-point representation of the raw acquisition record. The analysis record length can be increased at the expense of waveform update rate (see :SYSTem:PRECision:LENGth). The analysis record can be retrieved from any source.

If the <points_mode> is NORMal the analysis 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 analysis 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 analysis record and raw acquisition records.

NOTE

The decision about which record has the MAXimum number of points happens before any decimation that may result from a :WAVEform:POINts request. This means it is possible for the analysis record to have more points than the raw acquisition record, but then a :WAVEform:POINts request may cause an integer decimation that results in fewer points being returned by :WAVEform:DATA? than are in the raw acquisition record.

NOTE

If the :WAVEform:SOURce is not an analog or digital source, the only valid parameters for WAVEform:POINts:MODE is NORMal or MAXimum.

**Considerations for
MAXimum or RAW
data retrieval**

- The instrument must be stopped (see the :STOP command (see [page 279](#)) or the :DIGitize command (see [page 271](#)) in the root subsystem) in order to return more than the *analysis record*.
- :TIMEbase:MODE must be set to MAIN.

- :ACQUIRE:TYPE must be set to NORMAL or AVERAGE.
- MAXimum or RAW will allow up to the maximum memory depth number of 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 1555
 - [":WAVEform:DATA?"](#) on page 1563
 - [":ACQUIRE:POINTS\[:ANALog\]"](#) on page 297
 - [":SYSTem:PRECision:LENGth"](#) on page 1443
 - [":WAVEform:VIEW"](#) on page 1584
 - [":WAVEform:PREamble?"](#) on page 1570
 - [":WAVEform:POINTS"](#) on page 1566
 - [":TIMEbase:MODE"](#) on page 1472
 - [":ACQUIRE:TYPE"](#) on page 311
 - [":ACQUIRE:COUNT"](#) on page 288

:WAVEform:PREamble?

C (see [page 1708](#))

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.

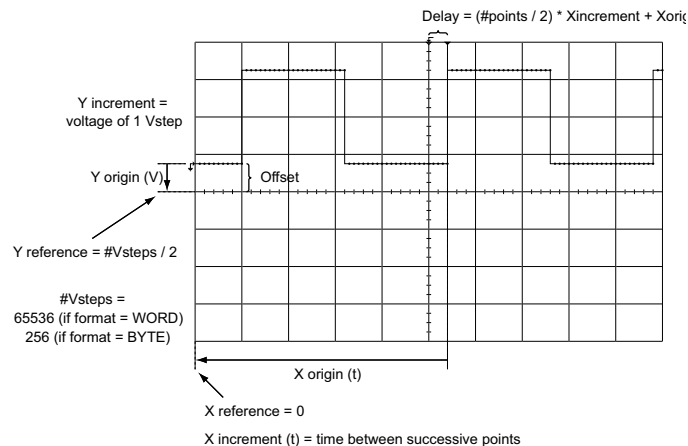
Return Format <preamble_block><NL>

```
<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> ::= 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 1555
 - [":ACQuire:COUNt"](#) on page 288
 - [":ACQuire:POINts\[:ANALog\]"](#) on page 297

- ":ACquire:TYPE" on page 311
- ":DIGitize" on page 271
- ":WAVEform:COUNt?" on page 1562
- ":WAVEform:DATA?" on page 1563
- ":WAVEform:FORMat" on page 1565
- ":WAVEform:POINts" on page 1566
- ":WAVEform:TYPE?" on page 1582
- ":WAVEform:XINCrement?" on page 1585
- ":WAVEform:XORigin?" on page 1586
- ":WAVEform:XREFerence?" on page 1587
- ":WAVEform:YINCrement?" on page 1588
- ":WAVEform:YORigin?" on page 1589
- ":WAVEform:YREFerence?" on page 1590

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.
'   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 49](#), “Programming Examples,” starting on page 1717

:WAVEform:SEGMented:ALL

N (see [page 1708](#))

Command Syntax :WAVEform:SEGMented:ALL {{0 | OFF} | {1 | ON}}

The :WAVEform:SEGMented:ALL command enables or disables the "waveform data for all segments" setting:

- ON – The :WAVEform:DATA? query returns data for all segments.
ON is available only when the :WAVEform:SOURce is an analog input channel.
When ON, the :WAVEform:DATA? query returns data for the number of segments multiplied by the number of points in a segment (which is the points value returned by the :WAVEform:POINts? or :WAVEform:PREamble? queries).
- OFF – The :WAVEform:DATA? query returns data for the current segment.

The ability to get waveform data for all segments is faster and more convenient than iterating over multiple segments and retrieving each one separately.

The performance improvement comes primarily when raw acquisition record data is being retrieved (:WAVEform:POINts:MODE RAW) instead of the analysis record data (:WAVEform:POINts:MODE NORMAl).

One corner case where you may want to retrieve the analysis record data is when the acquired data for a segment is less than the screen resolution and the analysis record is interpolated.

Most often though, the acquired data for a segment is enough to fill the screen, and less than the maximum analysis record size, so the segmented raw acquisition data and the analysis record data are the same.

If the number of segments is small, the raw acquisition segment size is greater than the analysis record size.

Query Syntax :WAVEform:SEGMented:ALL?

The :WAVEform:SEGMented:ALL? query returns the "waveform data for all segments" setting.

Return Format <setting><NL>

<setting> ::= {0 | 1}

- See Also**
- [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:SEGMented:XLISt?"](#) on page 1576
 - [":WAVEform:POINts:MODE"](#) on page 1568
 - [":WAVEform:POINts"](#) on page 1566
 - [":WAVEform:PREamble?"](#) on page 1570

:WAVEform:SEGMented:COUNT?

N (see [page 1708](#))

Query Syntax :WAVEform:SEGMented:COUNT?

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 296
 - [":ACQUIRE:SEGMented:COUNT"](#) on page 303
 - [":DIGitize"](#) on page 271
 - [":SINGLE"](#) on page 277
 - [":RUN"](#) on page 275
 - ["Introduction to :WAVEform Commands"](#) on page 1555

Example Code • ["Example Code"](#) on page 304

:WAVeform:SEGMented:TTAG?

N (see [page 1708](#))

Query Syntax :WAVeform:SEGMented:TTAG?

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 304
 - ["Introduction to :WAVeform Commands"](#) on page 1555

Example Code

- ["Example Code"](#) on page 304

:WAVEform:SEGMented:XLISt?

N (see [page 1708](#))

Query Syntax :WAVEform:SEGMented:XLISt? <xlist_type>

<xlist_type> ::= {RELXorigin | ABSXorigin | TTAG}

The :WAVEform:SEGMented:XLISt? query returns the X (time) information for all segments at once. The <xlist_type> option specifies the type of information that is returned:

- RELXorigin – The relative X-origin for each segment (the value returned by the :WAVEform:XORigin? query) is returned.
- TTAG – The time tag for each segment (the value returned by the :WAVEform:SEGMented:TTAG? query) will be returned.
- ABSXorigin – The sum of the values of the RELXorigin and TTAG types is returned for each segment.

This command is useful when getting the waveform data for all segments at once (see :WAVEform:SEGMented:ALL).

Return Format <return_value><NL>

<return_value> ::= binary block data in IEEE 488.2 # format, contains comma-separated string with X-info for all segments

- See Also**
- [":WAVEform:XORigin?"](#) on page 1586
 - [":WAVEform:SEGMented:TTAG?"](#) on page 1575
 - [":WAVEform:SEGMented:ALL"](#) on page 1573

:WAVEform:SOURce

C (see [page 1708](#))

Command Syntax :WAVEform:SOURce <source>

```
<source> ::= {CHANnel<n> | POD{1 | 2} | BUS{1 | 2} | FUNCTION<m>
             | MATH<m> | FFT | WMEMory<r> | {PBUS{1 | 2} | SBUS{1 | 2}}}
```

```
<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 1565).

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 <source><NL>

```
<source> ::= {CHAN<n> | POD{1 | 2} | BUS{1 | 2} | FUNC<m> | WMEM<r>
| PBUS{1 | 2}}
```

```
<n> ::= 1 to (# analog channels) in NR1 format
```

```
<m> ::= 1 to (# math functions) in NR1 format
```

```
<r> ::= 1 to (# ref waveforms) in NR1 format
```

- See Also**
- ["Introduction to :WAVEform Commands"](#) on page 1555
 - [":DIGitize"](#) on page 271
 - [":WAVEform:FORMat"](#) on page 1565
 - [":WAVEform:BYTeorder"](#) on page 1561
 - [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:PREamble?"](#) on page 1570

Example Code

```
' 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
' 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 ":WAVEform: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) - 1
    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 49](#), "Programming Examples," starting on page 1717

:WAVEform:SOURce:SUBSource

C (see [page 1708](#))

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 1555
 - [":WAVEform:SOURce"](#) on page 1577

:WAVEform:TYPE?

C (see [page 1708](#))

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 }

NOTE

If the :WAVEform:SOURce is POD1, POD2, or SBUS1, SBUS2, the type is always NORM.

-
- See Also**
- ["Introduction to :WAVEform Commands"](#) on page 1555
 - [":ACQUIRE:TYPE"](#) on page 311
 - [":WAVEform:DATA?"](#) on page 1563
 - [":WAVEform:PREamble?"](#) on page 1570
 - [":WAVEform:SOURce"](#) on page 1577

:WAVEform:UNSigned

C (see [page 1708](#))

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 1555
 - [":WAVEform:SOURce"](#) on page 1577

:WAVEform:VIEW

C (see [page 1708](#))

Command Syntax :WAVEform:VIEW <view>

<view> ::= {MAIN | ALL}

The :WAVEform:VIEW command sets the view setting associated with the currently selected waveform:

- MAIN – This view specifies the data you see in the oscilloscope's main waveform display area.
- ALL – Available only when Digitizer mode is on (see :ACQUIRE:DIGitizer), this view specifies all the captured data, which may extend beyond the edges of the oscilloscope's main waveform display area depending on the settings for sample rate, memory depth, and horizontal time/div.

Query Syntax :WAVEform:VIEW?

The :WAVEform:VIEW? query returns the view setting associated with the currently selected waveform.

Return Format <view><NL>

<view> ::= {MAIN | ALL}

- See Also**
- ["Introduction to :WAVEform Commands"](#) on page 1555
 - [":WAVEform:POINTs"](#) on page 1566
 - [":ACQUIRE:DIGitizer"](#) on page 290

:WAVEform:XINCrement?

C (see [page 1708](#))

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 1555
 - [":WAVEform:PREamble?"](#) on page 1570

- Example Code**
- ["Example Code"](#) on page 1571

:WAVeform:XORigin?

C (see [page 1708](#))

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 1555
 - [":WAVeform:PREamble?"](#) on page 1570
 - [":WAVeform:XREference?"](#) on page 1587

Example Code • ["Example Code"](#) on page 1571

:WAVeform:XREFerence?

C (see [page 1708](#))

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 1555
 - [":WAVeform:PREamble?"](#) on page 1570
 - [":WAVeform:XORigin?"](#) on page 1586

Example Code • ["Example Code"](#) on page 1571

:WAVeform:YINCrement?

C (see [page 1708](#))

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 1555
- [":WAVeform:PREamble?"](#) on page 1570

Example Code

- ["Example Code"](#) on page 1571

:WAVeform:YORigin?

C (see [page 1708](#))

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 1555
 - [":WAVeform:PREamble?"](#) on page 1570
 - [":WAVeform:YREference?"](#) on page 1590

Example Code • ["Example Code"](#) on page 1571

:WAVeform:YREference?

C (see [page 1708](#))

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 1555
 - [":WAVeform:PREamble?"](#) on page 1570
 - [":WAVeform:YORigin?"](#) on page 1589

Example Code • ["Example Code"](#) on page 1571

42 :WGEN<w> Commands

When the built-in waveform generator is licensed (WAVEGEN license), 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 1594.

Table 178 :WGEN<w> Commands Summary

Command	Query	Options and Query Returns
:WGEN<w>:ARbitrary:BYTeorder <order> (see page 1596)	:WGEN<w>:ARbitrary:BYTeorder? (see page 1596)	<order> ::= {MSBFirst LSBFirst} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARbitrary:DATA {<binary> <value>, <value> ...} (see page 1597)	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 to (# WaveGen outputs) in NR1 format
n/a	:WGEN<w>:ARbitrary:DATA:ATTRibute:POINTs? (see page 1600)	<points> ::= number of points in NR1 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARbitrary:DATA:CLEar (see page 1601)	n/a	<w> ::= 1 to (# WaveGen outputs) in NR1 format

Table 178 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:ARbitrary:DATA:DAC {<binary> <value>, <value> ...} (see page 1602)	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 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARbitrary:INTerpolate {{0 OFF} {1 ON}} (see page 1603)	:WGEN<w>:ARbitrary:INTerpolate? (see page 1603)	{0 1} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:ARbitrary:STORe <source> (see page 1604)	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 to (# WaveGen outputs) in NR1 format
:WGEN<w>:DCMode <mode> (see page 1605)	:WGEN<w>:DCMode? (see page 1605)	<mode> ::= {PREcise WIDerange}
:WGEN<w>:FREQuency <frequency> (see page 1606)	:WGEN<w>:FREQuency? (see page 1606)	<frequency> ::= frequency in Hz in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:FUNctIon <signal> (see page 1607)	:WGEN<w>:FUNctIon? (see page 1612)	<signal> ::= {SINusoid SQUare RAMP PULSe NOISe DC SINC EXPRise EXPFall CARDiac GAUSSian ARBitary} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:FUNctIon:PULSe:WIDTh <width> (see page 1613)	:WGEN<w>:FUNctIon:PULSe:WIDTh? (see page 1613)	<width> ::= pulse width in seconds in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format

Table 178 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:FUNCTION:SQU are:DCYCLE <percent> (see page 1615)	:WGEN<w>:FUNCTION:SQU are:DCYCLE? (see page 1615)	<percent> ::= duty cycle percentage in NR1 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:MODulation:A M:DEPTh <percent> (see page 1616)	:WGEN<w>:MODulation:A M:DEPTh? (see page 1616)	<percent> ::= AM depth percentage from 0% to 100% in NR1 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:A M:FREQuency <frequency> (see page 1617)	:WGEN<w>:MODulation:A M:FREQuency? (see page 1617)	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F M:DEVIation <frequency> (see page 1618)	:WGEN<w>:MODulation:F M:DEVIation? (see page 1618)	<frequency> ::= frequency deviation in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F M:FREQuency <frequency> (see page 1619)	:WGEN<w>:MODulation:F M:FREQuency? (see page 1619)	<frequency> ::= modulating waveform frequency in Hz in NR3 format <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:F UNCTION <shape> (see page 1620)	:WGEN<w>:MODulation:F UNCTION? (see page 1620)	<shape> ::= {SINusoid} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:N OISe <percent> (see page 1621)	:WGEN<w>:MODulation:N OISe? (see page 1621)	<percent> ::= 0 to 100 <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:S TATe {{0 OFF} {1 ON}} (see page 1622)	:WGEN<w>:MODulation:S TATe? (see page 1622)	{0 1} <w> ::= 1 in NR1 format
:WGEN<w>:MODulation:T YPE <type> (see page 1623)	:WGEN<w>:MODulation:T YPE? (see page 1623)	<type> ::= {AM FM} <w> ::= 1 in NR1 format
:WGEN<w>:OUTPut {{0 OFF} {1 ON}} (see page 1624)	:WGEN<w>:OUTPut? (see page 1624)	{0 1} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:OUTPut:BURSt (see page 1625)	n/a	n/a

Table 178 :WGEN<w> Commands Summary (continued)

Command	Query	Options and Query Returns
:WGEN<w>:OUTPut:BURSt :COUNT <number> (see page 1626)	:WGEN<w>:OUTPut:BURSt :COUNT? (see page 1626)	<w> ::= 1 to (# WaveGen outputs) in NR1 format <number> ::= 1 to 10000 in NR1 format
:WGEN<w>:OUTPut:LOAD <impedance> (see page 1627)	:WGEN<w>:OUTPut:LOAD? (see page 1627)	<impedance> ::= {ONEMeg FIFTy} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:OUTPut:MODE <mode> (see page 1628)	:WGEN<w>:OUTPut:MODE? (see page 1628)	<mode> ::= {NORMal BURSt}
:WGEN<w>:OUTPut:POLar ity <polarity> (see page 1629)	:WGEN<w>:OUTPut:POLar ity? (see page 1629)	<polarity> ::= {NORMal INVERTed} <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:PERiod <period> (see page 1630)	:WGEN<w>:PERiod? (see page 1630)	<period> ::= period in seconds in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:RST (see page 1631)	n/a	<w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage <amplitude> (see page 1632)	:WGEN<w>:VOLTage? (see page 1632)	<amplitude> ::= amplitude in volts in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:HIGH <high> (see page 1633)	:WGEN<w>:VOLTage:HIGH ? (see page 1633)	<high> ::= high-level voltage in volts, in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:LOW <low> (see page 1634)	:WGEN<w>:VOLTage:LOW? (see page 1634)	<low> ::= low-level voltage in volts, in NR3 format <w> ::= 1 to (# WaveGen outputs) in NR1 format
:WGEN<w>:VOLTage:OFFS et <offset> (see page 1635)	:WGEN<w>:VOLTage:OFFS et? (see page 1635)	<offset> ::= offset in volts in NR3 format <w> ::= 1 to (# WaveGen outputs) 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.

```
:WGEN:FUNC SIN;OUTP 0;FREQ +1.00000E+03;VOLT +500.0E-03;  
VOLT:OFFS +0.0E+00;:WGEN:OUTP:LOAD ONEM
```

:WGEN<w>:ARbitrary:BYTeorder

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:BYTeorder <order>

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":WGEN<w>:ARbitrary:DATA:DAC"](#) on page 1602

:WGEN<w>:ARbitrary:DATA

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:DATA {<binary> | <value>, <value> ...}

<w> ::= 1 to (# WaveGen outputs) 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 1602
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944
 - ["Commands Not Supported in Multiple Program Message Units"](#) on page 1713

Example Code ' Waveform generator arbitrary data commands example.

```
' -----

Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
    Private Declare PtrSafe Sub CopyMemory Lib "kernel32" Alias "RtlMove
Memory" _
        (dest As Any, source As Any, ByVal bytes As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Lo
ng)
    Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory"
_
        (dest As Any, source As Any, ByVal bytes As Long)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
myScope.IO.Clear ' Clear the interface.
myScope.IO.Timeout = 20000 ' Set I/O communication timeout.

' Turn on arbitrary waveform generator function.
myScope.WriteString ":WGEN1:OUTPut ON"
myScope.WriteString ":WGEN1:FUNCTION ARbitrary"
myScope.WriteString ":WGEN1:FUNCTION?"
strQueryResult = myScope.ReadString
Debug.Print "WaveGen1 function: " + strQueryResult

DefaultArbitraryWaveform

' Download comma-separated floating-point values.
myScope.WriteString ":WGEN1:ARbitrary:DATA 0.0, 0.5, 1.0, 0.5, 0.0, -0
.5, -1.0, -0.5"
Debug.Print "WaveGen1 CSV floating-point values downloaded."
Sleep 5000

DefaultArbitraryWaveform

' Download comma-separated 16-bit integer (DAC) values.
myScope.WriteString ":WGEN1:ARbitrary:DATA:DAC 0, 255, 511, 255, 0, -2
56, -512, -256"
Debug.Print "WaveGen1 CSV 16-bit integer (DAC) values downloaded."
Sleep 5000

' Set the byte order for binary data.
myScope.WriteString ":WGEN1:ARbitrary:BYTeorder LSBFirst"
myScope.WriteString ":WGEN1:ARbitrary:BYTeorder?"
strQueryResult = myScope.ReadString
Debug.Print "WaveGen1 byte order for binary data: " + strQueryResult

DefaultArbitraryWaveform

' Download binary floating-point values.
Dim mySingleArray(8) As Single
mySingleArray(0) = 0!
mySingleArray(1) = 0.5!
mySingleArray(2) = 1!
mySingleArray(3) = 0.5!
mySingleArray(4) = 0!
mySingleArray(5) = -0.5!
mySingleArray(6) = -1!
mySingleArray(7) = -0.5!

Dim myByteArray(32) As Byte
CopyMemory myByteArray(0), mySingleArray(0), 32 * LenB(myByteArray(0))

myScope.WriteIEEEBlock ":WGEN1:ARbitrary:DATA", myByteArray, True
Debug.Print "WaveGen1 binary floating-point values downloaded."
Sleep 5000

DefaultArbitraryWaveform

' Download binary 16-bit integer (DAC) values.

```

```

Dim myIntegerArray(8) As Integer
myIntegerArray(0) = 0
myIntegerArray(1) = 255
myIntegerArray(2) = 511
myIntegerArray(3) = 255
myIntegerArray(4) = 0
myIntegerArray(5) = -256
myIntegerArray(6) = -512
myIntegerArray(7) = -256

Dim myByteArray2(16) As Byte
CopyMemory myByteArray2(0), myIntegerArray(0), 16 * LenB(myByteArray2(
0))

myScope.WriteIEEEBlock ":WGEN1:ARbitrary:DATA:DAC", myByteArray2, True
Debug.Print "WaveGen1 binary 16-bit integer (DAC) values downloaded."
Sleep 5000

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

'
' Initialize WaveGen1 to a known state.
' -----
Private Sub DefaultArbitraryWaveform()

On Error GoTo VisaComError

' Load default arbitrary waveform.
myScope.WriteString ":WGEN1:ARbitrary:DATA:CLear"
Debug.Print "WaveGen1 default arbitrary waveform loaded."
Sleep 5000

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description
End

End Sub

```

:WGEN<w>:ARbitrary:DATA:ATTRibute:POINts?

N (see [page 1708](#))

Query Syntax :WGEN<w>:ARbitrary:DATA:ATTRibute:POINts?

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":WGEN<w>:ARbitrary:DATA:DAC"](#) on page 1602
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944

:WGEN<w>:ARbitrary:DATA:CLEar

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:DATA:CLEar

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":WGEN<w>:ARbitrary:DATA:DAC"](#) on page 1602
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944

- Example Code**
- ["Example Code"](#) on page 1597

:WGEN<w>:ARbitrary:DATA:DAC

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:DATA:DAC {<binary> | <value>, <value> ...}

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944

- Example Code**
- ["Example Code"](#) on page 1597

:WGEN<w>:ARbitrary:INterpolate

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:INterpolate {{0 | OFF} | {1 | ON}}

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":WGEN<w>:ARbitrary:DATA:DAC"](#) on page 1602
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944

:WGEN<w>:ARbitrary:STORe

N (see [page 1708](#))

Command Syntax :WGEN<w>:ARbitrary:STORe <source>

<w> ::= 1 to (# WaveGen outputs) 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 1597
 - [":WGEN<w>:ARbitrary:DATA:DAC"](#) on page 1602
 - [":SAVE:ARbitrary\[:START\]"](#) on page 957
 - [":RECall:ARbitrary\[:START\]"](#) on page 944

:WGEN<w>:DCMode

N (see [page 1708](#))

Command Syntax :WGEN<w>:DCMode <mode>

<mode> ::= {PRECise | WIDerange}

When the DC waveform type is selected (see :WGEN<w>:FUNCtion), the :WGEN<w>:DCMode command specifies the DC output mode:

- PRECise – You can select from a smaller range of output voltages (–1 V to 1 V) with greater accuracy of the specified level.
- WIDerange – You can select from a wider range of output voltages (–8 V to 8 V) with with less accuracy of the specified level.

Query Syntax :WGEN<w>:DCMode?

The :WGEN<w>:DCMode? query returns the selected DC output mode selection.

Return Format <mode><NL>

<mode> ::= {PREC | WID}

See Also • [":WGEN<w>:FUNCtion"](#) on page 1607

:WGEN<w>:FREQuency

N (see [page 1708](#))

Command Syntax :WGEN<w>:FREQuency <frequency>

<w> ::= 1 to (# WaveGen outputs) 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 1594
 - [":WGEN<w>:FUNction"](#) on page 1607
 - [":WGEN<w>:PERiod"](#) on page 1630

:WGEN<w>:FUNCTION

N (see [page 1708](#))

Command Syntax :WGEN<w>:FUNCTION <signal>

<w> ::= 1 to (# WaveGen outputs) 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 ¹	Max. Amplitude (High-Z) ²	Offset Range (High-Z) ²
SINusoid	Use these commands to set the sine signal parameters: ▪ ":WGEN<w>:FREQuency" on page 1606 ▪ ":WGEN<w>:PERiod" on page 1630 ▪ ":WGEN<w>:VOLTage" on page 1632 ▪ ":WGEN<w>:VOLTage:OFFSet" on page 1635 ▪ ":WGEN<w>:VOLTage:HIGH" on page 1633 ▪ ":WGEN<w>:VOLTage:LOW" on page 1634	10 MHz to 100 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)
SQUare	Use these commands to set the square wave signal parameters: ▪ ":WGEN<w>:FREQuency" on page 1606 ▪ ":WGEN<w>:PERiod" on page 1630 ▪ ":WGEN<w>:VOLTage" on page 1632 ▪ ":WGEN<w>:VOLTage:OFFSet" on page 1635 ▪ ":WGEN<w>:VOLTage:HIGH" on page 1633 ▪ ":WGEN<w>:VOLTage:LOW" on page 1634 ▪ ":WGEN<w>:FUNCTION:SQUare:DCYCLe" on page 1615 The duty cycle can be adjusted from 0.5% to 99.5% depending on the square wave frequency. The resolution of the percentages that can be selected is finer at lower frequencies and coarser at higher frequencies.	10 MHz to 50 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)

Waveform Type	Characteristics	Frequency Range ¹	Max. Amplitude (High-Z) ²	Offset Range (High-Z) ²
RAMP	Use these commands to set the ramp signal parameters: ▪ ":WGEN<w>:FREQuency" on page 1606 ▪ ":WGEN<w>:PERiod" on page 1630 ▪ ":WGEN<w>:VOLTage" on page 1632 ▪ ":WGEN<w>:VOLTage:OFFSet" on page 1635 ▪ ":WGEN<w>:VOLTage:HIGH" on page 1633 ▪ ":WGEN<w>:VOLTage:LOW" on page 1634 ▪ ":WGEN<w>:FUNCTion:RAMP:SYMMetry" on page 1614 Symmetry represents the amount of time per cycle that the ramp waveform is rising and can be adjusted from 0% to 100%.	300 MHz to 5 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)
PULSe	Use these commands to set the pulse signal parameters: ▪ ":WGEN<w>:FREQuency" on page 1606 ▪ ":WGEN<w>:PERiod" on page 1630 ▪ ":WGEN<w>:VOLTage" on page 1632 ▪ ":WGEN<w>:VOLTage:OFFSet" on page 1635 ▪ ":WGEN<w>:VOLTage:HIGH" on page 1633 ▪ ":WGEN<w>:VOLTage:LOW" on page 1634 ▪ ":WGEN<w>:FUNCTion:PULSe:WIDTh" on page 1613 The minimum pulse width is 10 ns at all frequencies. The pulse width can be adjusted in 5 ns increments from the minimum to the period minus the minimum.	10 MHz to 50 MHz.	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)

Waveform Type	Characteristics	Frequency Range ¹	Max. Amplitude (High-Z) ²	Offset Range (High-Z) ²
DC	Use this command to set the DC level: ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635 Use this command to set the DC output mode: ▪ <code>":WGEN<w>:DCMode"</code> on page 1605	n/a	n/a	±8.00 V or ±1.00 V, depending on the DC output mode selection
NOISe	Use these commands to set the noise signal parameters: ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635 ▪ <code>":WGEN<w>:VOLTage:HIGH"</code> on page 1633 ▪ <code>":WGEN<w>:VOLTage:LOW"</code> on page 1634	n/a	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)
SINC	Use these commands to set the sine cardinal signal parameters: ▪ <code>":WGEN<w>:FREQuency"</code> on page 1606 ▪ <code>":WGEN<w>:PERiod"</code> on page 1630 ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635	300 mHz to 1 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(400 mV - 0.25 * amplitude) When amplitude ≥50 mV: ±(4 V - 0.25 * amplitude)
EXPRise	Use these commands to set the exponential rise signal parameters: ▪ <code>":WGEN<w>:FREQuency"</code> on page 1606 ▪ <code>":WGEN<w>:PERiod"</code> on page 1630 ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635 ▪ <code>":WGEN<w>:VOLTage:HIGH"</code> on page 1633 ▪ <code>":WGEN<w>:VOLTage:LOW"</code> on page 1634	300 mHz to 5 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: ±(800 mV - 0.5 * amplitude) When amplitude ≥50 mV: ±(8 V - 0.5 * amplitude)

Waveform Type	Characteristics	Frequency Range ¹	Max. Amplitude (High-Z) ²	Offset Range (High-Z) ²
EXPfall	Use these commands to set the exponential fall signal parameters: ▪ <code>":WGEN<w>:FREQuency"</code> on page 1606 ▪ <code>":WGEN<w>:PERiod"</code> on page 1630 ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635 ▪ <code>":WGEN<w>:VOLTage:HIGH"</code> on page 1633 ▪ <code>":WGEN<w>:VOLTage:LOW"</code> on page 1634	300 mHz to 5 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: $\pm(800 \text{ mV} - 0.5 * \text{amplitude})$ When amplitude $\geq 50 \text{ mV}$: $\pm(8 \text{ V} - 0.5 * \text{amplitude})$
CARDiac	Use these commands to set the cardiac signal parameters: ▪ <code>":WGEN<w>:FREQuency"</code> on page 1606 ▪ <code>":WGEN<w>:PERiod"</code> on page 1630 ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635	300 mHz to 2.5 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: $\pm(400 \text{ mV} - 0.25 * \text{amplitude})$ When amplitude $\geq 50 \text{ mV}$: $\pm(4 \text{ V} - 0.25 * \text{amplitude})$
GAUSSian	Use these commands to set the gaussian pulse signal parameters: ▪ <code>":WGEN<w>:FREQuency"</code> on page 1606 ▪ <code>":WGEN<w>:PERiod"</code> on page 1630 ▪ <code>":WGEN<w>:VOLTage"</code> on page 1632 ▪ <code>":WGEN<w>:VOLTage:OFFSet"</code> on page 1635	300 mHz to 5 MHz	2 mVpp to 8 Vpp	When amplitude <50 mV: $\pm(400 \text{ mV} - 0.25 * \text{amplitude})$ When amplitude $\geq 50 \text{ mV}$: $\pm(4 \text{ V} - 0.25 * \text{amplitude})$

Waveform Type	Characteristics	Frequency Range ¹	Max. Amplitude (High-Z) ²	Offset Range (High-Z) ²
ARbitrary	Use these commands to set the arbitrary signal parameters: ▪ ":WGEN<w>:FREQuency" on page 1606 ▪ ":WGEN<w>:PERiod" on page 1630 ▪ ":WGEN<w>:VOLTage" on page 1632 ▪ ":WGEN<w>:VOLTage:OFFSet" on page 1635 ▪ ":WGEN<w>:VOLTage:HIGH" on page 1633 ▪ ":WGEN<w>:VOLTage:LOW" on page 1634	300 mHz to 5 MHz	2 mVpp to 10 Vpp	When amplitude <50 mV: $\pm(800 \text{ mV} - 0.5 * \text{amplitude})$ When amplitude $\geq 50 \text{ mV}$: $\pm(8 \text{ V} - 0.5 * \text{amplitude})$
¹ When burst output mode is selected (see ":WGEN<w>:OUTPut:MODE" on page 1628), the maximum frequency is limited to 5 MHz. ² When the output load is 50 Ω , these values are halved.				

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 1594
- **":WGEN<w>:MODulation:NOISe"** on page 1621

:WGEN<w>:FUNCTION:PULSE:WIDTH

N (see [page 1708](#))

Command Syntax :WGEN<w>:FUNCTION:PULSE:WIDTH <width>

<w> ::= 1 to (# WaveGen outputs) 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607

:WGEN<w>:FUNCTION:RAMP:SYMMetry

N (see [page 1708](#))

Command Syntax :WGEN<w>:FUNCTION:RAMP:SYMMetry <percent>

<w> ::= 1 to (# WaveGen outputs) in NR1 format

<percent> ::= symmetry percentage from 0% to 100% in NR1 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 NR1 format

- See Also**
- ["Introduction to :WGEN<w> Commands"](#) on page 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607

:WGEN<w>:FUNCTION:SQUare:DCYCLE

N (see [page 1708](#))

Command Syntax :WGEN<w>:FUNCTION:SQUare:DCYCLE <percent>

<w> ::= 1 to (# WaveGen outputs) in NR1 format

<percent> ::= duty cycle percentage in NR1 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.

The duty cycle percentage can be from 0.5% to 99.5% depending on the square wave frequency (:WGEN<w>:FREQuency). The resolution of the percentages that can be selected is finer at lower frequencies and coarser at higher frequencies.

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 in NR1 format

- See Also**
- ["Introduction to :WGEN<w> Commands"](#) on page 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607

:WGEN<w>:MODulation:AM:DEPT

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:AM:DEPT <percent>

<w> ::= 1 in NR1 format

<percent> ::= AM depth percentage from 0% to 100% in NR1 format

The :WGEN<w>:MODulation:AM:DEPT 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.

Query Syntax :WGEN<w>:MODulation:AM:DEPT?

The :WGEN<w>:MODulation:AM:DEPT? query returns the AM depth percentage setting.

Return Format <percent><NL>

<percent> ::= AM depth percentage from 0% to 100% in NR1 format

- See Also**
- [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:DEViation"](#) on page 1618
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:FUNCTion"](#) on page 1620
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:AM:FREQuency

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:AM:FREQuency <frequency>

<w> ::= 1 in NR1 format

<frequency> ::= modulating waveform frequency in Hz in NR3 format

The :WGEN<w>:MODulation:AM:FREQuency command specifies the frequency of the modulating signal.

Query Syntax :WGEN<w>:MODulation:AM:FREQuency?

The :WGEN<w>:MODulation:AM:FREQuency? query returns the frequency of the modulating signal.

Return Format <frequency><NL>

<frequency> ::= modulating waveform frequency in Hz in NR3 format

- See Also**
- [":WGEN<w>:MODulation:AM:DEPTH"](#) on page 1616
 - [":WGEN<w>:MODulation:FM:DEViation"](#) on page 1618
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:FUNCTion"](#) on page 1620
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:FM:DEVIation

N (see [page 1708](#))

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 1616
 - [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:FUNcTion"](#) on page 1620
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:FM:FREQuency

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:FM:FREQuency <frequency>

<w> ::= 1 in NR1 format

<frequency> ::= modulating waveform frequency in Hz in NR3 format

The :WGEN<w>:MODulation:FM:FREQuency command specifies the frequency of the modulating signal.

Query Syntax :WGEN<w>:MODulation:FM:FREQuency?

The :WGEN<w>:MODulation:FM:FREQuency? query returns the frequency of the modulating signal.

Return Format <frequency><NL>

<frequency> ::= modulating waveform frequency in Hz in NR3 format

- See Also**
- [":WGEN<w>:MODulation:AM:DEPTH"](#) on page 1616
 - [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:DEViation"](#) on page 1618
 - [":WGEN<w>:MODulation:FUNCTion"](#) on page 1620
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:FUNCTion

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:FUNCTion <shape>

<w> ::= 1 in NR1 format

<shape> ::= {SINusoid}

The :WGEN<w>:MODulation:FUNCTion command specifies the shape of the modulating signal.

This command applies to AM and FM modulation.

Query Syntax :WGEN<w>:MODulation:FUNCTion?

The :WGEN<w>:MODulation:FUNCTion? query returns the specified modulating signal shape.

Return Format <shape><NL>

<shape> ::= {SIN}

- See Also**
- [":WGEN<w>:MODulation:AM:DEPTH"](#) on page 1616
 - [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:DEViation"](#) on page 1618
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:NOISe

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:NOISe <percent>

<w> ::= 1 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 AUX 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 1607

:WGEN<w>:MODulation:STATe

N (see [page 1708](#))

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 1616
 - [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:DEViation"](#) on page 1618
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:FUNCTion"](#) on page 1620
 - [":WGEN<w>:MODulation:TYPE"](#) on page 1623

:WGEN<w>:MODulation:TYPE

N (see [page 1708](#))

Command Syntax :WGEN<w>:MODulation:TYPE <type>

<w> ::= 1 in NR1 format

<type> ::= {AM | FM}

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.

Query Syntax :WGEN<w>:MODulation:TYPE?

The :WGEN<w>:MODulation:TYPE? query returns the selected modulation type.

Return Format <type><NL>

<type> ::= {AM | FM}

- See Also**
- [":WGEN<w>:MODulation:AM:DEPTH"](#) on page 1616
 - [":WGEN<w>:MODulation:AM:FREQuency"](#) on page 1617
 - [":WGEN<w>:MODulation:FM:DEVIation"](#) on page 1618
 - [":WGEN<w>:MODulation:FM:FREQuency"](#) on page 1619
 - [":WGEN<w>:MODulation:FUNCTion"](#) on page 1620
 - [":WGEN<w>:MODulation:STATe"](#) on page 1622

:WGEN<w>:OUTPut

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut <on_off>

<w> ::= 1 to (# WaveGen outputs) 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 1594

:WGEN<w>:OUTPut:BURSt

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut:BURSt

When the BURSt output mode is selected (by the :WGEN<w>:OUTPut:MODE command), the :WGEN<w>:OUTPut:BURSt command causes the :WGEN<w>:OUTPut:BURSt:COUNT number of cycles 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 1628
 - [":WGEN<w>:OUTPut:BURSt:COUNT"](#) on page 1626

:WGEN<w>:OUTPut:BURSt:COUNT

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut:BURSt:COUNT <number>

<w> ::= 1 to (# WaveGen outputs) in NR1 format

<number> ::= 1 to 10000 in NR1 format

When the BURSt output mode is selected (by the :WGEN<w>:OUTPut:MODE command), the :WGEN<w>:OUTPut:BURSt:COUNT command sets the burst count number.

Query Syntax :WGEN<w>:OUTPut:BURSt:COUNT?

The :WGEN<w>:OUTPut:BURSt:COUNT? query returns the currently set burst count number.

Return Format <number><NL>

<number> ::= 1 to 10000 in NR1 format

- See Also**
- [":WGEN<w>:OUTPut:MODE"](#) on page 1628
 - [":WGEN<w>:OUTPut:BURSt"](#) on page 1625

:WGEN<w>:OUTPut:LOAD

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut:LOAD <impedance>

<w> ::= 1 to (# WaveGen outputs) 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 1594

:WGEN<w>:OUTPut:MODE

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut:MODE <mode>

<mode> ::= {NORMal | BURSt}

The :WGEN<w>:OUTPut:MODE command specifies whether the defined waveform is output continuously or as a specified number of cycles (burst):

- NORMal – the defined waveform is output continuously.
- BURSt – the :WGEN<w>:OUTPut:BURSt:COUNt number of cycles of the defined waveform is output when you send the :WGEN<w>:OUTPut:BURSt command.

When burst output mode is selected, the maximum frequency is limited to 5 MHz.

Query Syntax :WGEN<w>:OUTPut:MODE?

The :WGEN<w>:OUTPut:MODE? query returns the output mode setting.

Return Format <mode><NL>

<mode> ::= {NORMal | BURSt}

- See Also**
- [":WGEN<w>:OUTPut:BURSt"](#) on page 1625
 - [":WGEN<w>:OUTPut:BURSt:COUNt"](#) on page 1626

:WGEN<w>:OUTPut:POLarity

N (see [page 1708](#))

Command Syntax :WGEN<w>:OUTPut:POLarity <polarity>

<w> ::= 1 to (# WaveGen outputs) 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 1594

:WGEN<w>:PERiod

N (see [page 1708](#))

Command Syntax :WGEN<w>:PERiod <period>

<w> ::= 1 to (# WaveGen outputs) 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 1594
 - [":WGEN<w>:FUNction"](#) on page 1607
 - [":WGEN<w>:FREQuency"](#) on page 1606

:WGEN<w>:RST

N (see [page 1708](#))

Command Syntax :WGEN<w>:RST

<w> ::= 1 to (# WaveGen outputs) 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607
 - [":WGEN<w>:FREQUENCY"](#) on page 1606

:WGEN<w>:VOLTage

N (see [page 1708](#))

Command Syntax :WGEN<w>:VOLTage <amplitude>

<w> ::= 1 to (# WaveGen outputs) 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 ± 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607
 - [":WGEN<w>:VOLTage:OFFSet"](#) on page 1635
 - [":WGEN<w>:VOLTage:HIGH"](#) on page 1633
 - [":WGEN<w>:VOLTage:LOW"](#) on page 1634

:WGEN<w>:VOLTage:HIGH

N (see [page 1708](#))

Command Syntax :WGEN<w>:VOLTage:HIGH <high>

<w> ::= 1 to (# WaveGen outputs) 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 ± 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607
 - [":WGEN<w>:VOLTage:LOW"](#) on page 1634
 - [":WGEN<w>:VOLTage"](#) on page 1632
 - [":WGEN<w>:VOLTage:OFFSet"](#) on page 1635

:WGEN<w>:VOLTage:LOW

N (see [page 1708](#))

Command Syntax :WGEN<w>:VOLTage:LOW <low>

<w> ::= 1 to (# WaveGen outputs) 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 ± 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607
 - [":WGEN<w>:VOLTage:LOW"](#) on page 1634
 - [":WGEN<w>:VOLTage"](#) on page 1632
 - [":WGEN<w>:VOLTage:OFFSet"](#) on page 1635

:WGEN<w>:VOLTage:OFFSet

N (see [page 1708](#))

Command Syntax :WGEN<w>:VOLTage:OFFSet <offset>

<w> ::= 1 to (# WaveGen outputs) 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 ± 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 1594
 - [":WGEN<w>:FUNCTION"](#) on page 1607
 - [":WGEN<w>:VOLTage"](#) on page 1632
 - [":WGEN<w>:VOLTage:HIGH"](#) on page 1633
 - [":WGEN<w>:VOLTage:LOW"](#) on page 1634

43 :WMEMemory<r> Commands

Control reference waveforms.

Table 179 :WMEMemory<r> Commands Summary

Command	Query	Options and Query Returns
:WMEMemory<r>:CLEar (see page 1639)	n/a	<r> ::= 1 to (# ref waveforms) in NR1 format
:WMEMemory<r>:DISPlay { {0 OFF} {1 ON} } (see page 1640)	:WMEMemory<r>:DISPlay? (see page 1640)	<r> ::= 1 to (# ref waveforms) in NR1 format {0 1}
:WMEMemory<r>:LABel <string> (see page 1641)	:WMEMemory<r>:LABel? (see page 1641)	<r> ::= 1 to (# ref waveforms) in NR1 format <string> ::= any series of 32 or less ASCII characters enclosed in quotation marks
:WMEMemory<r>:SAVE <source> (see page 1642)	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: Math functions whose x-axis is not frequency can be saved as reference waveforms.
:WMEMemory<r>:SKEW <skew> (see page 1643)	:WMEMemory<r>:SKEW? (see page 1643)	<r> ::= 1 to (# ref waveforms) in NR1 format <skew> ::= time in seconds in NR3 format

Table 179 :WMEemory<r> Commands Summary (continued)

Command	Query	Options and Query Returns
:WMEemory<r>:YOFFset <offset>[suffix] (see page 1644)	:WMEemory<r>:YOFFset? (see page 1644)	<r> ::= 1 to (# ref waveforms) in NR1 format <offset> ::= vertical offset value in NR3 format [suffix] ::= {V mV}
:WMEemory<r>:YRANge <range> (see page 1645)	:WMEemory<r>:YRANge? (see page 1645)	<r> ::= 1 to (# ref waveforms) in NR1 format <range> ::= vertical full-scale range value in NR3 format
:WMEemory<r>:YSCale <scale> (see page 1646)	:WMEemory<r>:YSCale? (see page 1646)	<r> ::= 1 to (# ref waveforms) in NR1 format <scale> ::= vertical units per division value in NR3 format

:WMEemory<r>:CLEar

N (see [page 1708](#))

Command Syntax :WMEemory<r>:CLEar

<r> ::= 1 to (# ref waveforms) in NR1 format

The :WMEemory<r>:CLEar command clears the specified reference waveform location.

- See Also**
- [Chapter 43](#), “:WMEemory<r> Commands,” starting on page 1637
 - [":WMEemory<r>:SAVE"](#) on page 1642
 - [":WMEemory<r>:DISPlay"](#) on page 1640

:WMEemory<r>:DISPlay

N (see [page 1708](#))

Command Syntax :WMEemory<r>:DISPlay <on_off>

<r> ::= 1 to (# ref waveforms) in NR1 format

<on_off> ::= {{1 | ON} | {0 | OFF}}

The :WMEemory<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 :WMEemory1:DISPlay is ON, sending the :WMEemory2:DISPlay ON command will automatically set :WMEemory1:DISPlay OFF.

Query Syntax :WMEemory<r>:DISPlay?

The :WMEemory<r>:DISPlay? query returns the current display setting for the reference waveform.

Return Format <on_off><NL>

<on_off> ::= {1 | 0}

- See Also**
- [Chapter 43](#), “:WMEemory<r> Commands,” starting on page 1637
 - [":WMEemory<r>:CLEar"](#) on page 1639
 - [":WMEemory<r>:LABel"](#) on page 1641

:WMEemory<r>:LABel

N (see [page 1708](#))

Command Syntax :WMEemory<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 :WMEemory<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 :WMEemory<r>:LABel?

The :WMEemory<r>:LABel? query returns the label associated with a particular reference waveform.

Return Format <string><NL>
 <string> ::= quoted ASCII string

See Also

- [Chapter 43](#), “:WMEemory<r> Commands,” starting on page 1637
- [":WMEemory<r>:DISPlay"](#) on page 1640

:WMEemory<r>:SAVE

N (see [page 1708](#))

Command Syntax :WMEemory<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 :WMEemory<r>:SAVE command copies the analog channel or math function waveform to the specified reference waveform location.

NOTE

Math functions whose x-axis is not frequency can be saved as reference waveforms.

-
- See Also**
- [Chapter 43](#), “:WMEemory<r> Commands,” starting on page 1637
 - [":WMEemory<r>:DISPlay"](#) on page 1640

:WMEemory<r>:SKEW

N (see [page 1708](#))

Command Syntax :WMEemory<r>:SKEW <skew>

<r> ::= 1 to (# ref waveforms) in NR1 format

<skew> ::= time in seconds in NR3 format

The :WMEemory<r>:SKEW command sets the skew factor for the specified reference waveform.

Query Syntax :WMEemory<r>:SKEW?

The :WMEemory<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 43](#), “:WMEemory<r> Commands,” starting on page 1637
 - [":WMEemory<r>:DISPlay"](#) on page 1640
 - [":WMEemory<r>:YOFFset"](#) on page 1644
 - [":WMEemory<r>:YRANge"](#) on page 1645
 - [":WMEemory<r>:YSCale"](#) on page 1646

:WMEMory<r>:YOFFset

N (see [page 1708](#))

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 43](#), “:WMEMory<r> Commands,” starting on page 1637
 - [":WMEMory<r>:DISPlay"](#) on page 1640
 - [":WMEMory<r>:YRANge"](#) on page 1645
 - [":WMEMory<r>:YSCale"](#) on page 1646
 - [":WMEMory<r>:SKEW"](#) on page 1643

:WMEemory<r>:YRANge

N (see [page 1708](#))

Command Syntax :WMEemory<r>:YRANge <range>

<r> ::= 1 to (# ref waveforms) in NR1 format

<range> ::= vertical full-scale range value in NR3 format

The :WMEemory<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 :WMEemory<r>:YRANge?

The :WMEemory<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 43](#), “:WMEemory<r> Commands,” starting on page 1637
 - [":WMEemory<r>:DISPlay"](#) on page 1640
 - [":WMEemory<r>:YOFFset"](#) on page 1644
 - [":WMEemory<r>:SKEW"](#) on page 1643
 - [":WMEemory<r>:YSCale"](#) on page 1646

:WMEMory<r>:YScale

N (see [page 1708](#))

Command Syntax :WMEMory<r>:YScale <scale>

<r> ::= 1 to (# ref waveforms) in NR1 format

<scale> ::= vertical units per division in NR3 format

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 43](#), “:WMEMory<r> Commands,” starting on page 1637
 - [":WMEMory<r>:DISPlay"](#) on page 1640
 - [":WMEMory<r>:YOFFset"](#) on page 1644
 - [":WMEMory<r>:YRANge"](#) on page 1645
 - [":WMEMory<r>:SKEW"](#) on page 1643

44 Obsolete and Discontinued Commands

Obsolete 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 1708).

Obsolete Command	Current Command Equivalent	Behavior Differences
:CHANnel<n>:BWLimit (see page 1656)	:CHANnel<n>:BANDwidth (see page 349) 20E+06	In earlier versions of the HD3-Series software, the :CHANnel<n>:BWLimit command mapped to a 40 MHz bandwidth limit on the channel. Now that the "global" :ACquire:BANDwidth command has been replaced by the "per channel" :CHANnel<n>:BANDwidth command, the :CHANnel<n>:BWLimit returns to the legacy 20 MHz limit of previous InfiniiVision oscilloscopes.
:TRIGger:DElay:TDElay:TIME (see page 1660)	:TRIGger:DElay:TDElay:TMIN (see page 1502)	The obsolete command was a delay time before looking for the trigger edge. The current command equivalent specifies the minimum time of a range in which to look for the trigger edge.

Discontinued Commands Discontinued commands are commands that were used by previous oscilloscopes, but are not supported by the InfiniiVision HD3-Series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

Discontinued Command	Current Command Equivalent	Comments
:ACQuire:AALias?	none	Anti-aliasing is always present in the HD3-Series oscilloscopes.
:ACQuire:BANDwidth	:CHANnel<n>:BANDwidth (see page 349)	In earlier versions of the HD3-Series software, the :ACQuire:BANDwidth command was a "global" bandwidth setting. The :CHANnel<n>:BANDwidth command is a "per channel" bandwidth setting (although its 5 MHz and 10 MHz settings are still global and must be applied to all channels).
:ACQuire:DAALias	none	There is no option to turn off anti-aliasing in the HD3-Series oscilloscopes.
:CHANnel2:SKEW	:CHANnel<n>:PROBe:SKEW (see page 369)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:ACTivity	:ACTivity (see page 262)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:LABel	:CHANnel<n>:LABel (see page 354) or :DIGital<n>:LABel (see page 401)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel:THReshold	:POD<n>:THReshold (see page 835) or :DIGital<d>:THReshold (see page 404)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel<n>:INPut	:CHANnel<n>:IMPedance (see page 352)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:CHANnel<n>:PMODE	none	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:DIGital<d>:POSition	:DIGital:ORDer[:SET] (see page 399)	The notion of digital channel position locations is not present in the HD3-Series oscilloscopes. You can, however, order digital channels using the :DIGital:ORDer[:SET] (see page 399) command.
:DISPlay:ANNotation<n>:X1Pos ition	:DISPlay:ANNotation<n>:XPOSi tion (see page 417)	Discontinued command positioned by pixel position, current command positions by percent of grid width.
:DISPlay:ANNotation<n>:Y1Pos ition	:DISPlay:ANNotation<n>:YPOSi tion (see page 418)	Discontinued command positioned by pixel position, current command positions by percent of grid height.
:DISPlay:CONNEct	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:DISPlay:GRATicule:TYPE	none	This command was for the TV trigger mode which is not available in the HD3-Series oscilloscopes.
:DISPlay:MENU	none	On earlier InfiniiVision oscilloscopes, this command would display a particular softkey menu. The HD3-Series oscilloscopes do not have softkey menus.
:DISPlay:ORDer	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:DISPlay:SIDebar	none	On earlier InfiniiVision oscilloscopes, this command would display a particular sidebar dialog box. The HD3-Series oscilloscopes do not have sidebar dialog boxes.
:DISPlay:VECTors	none	On the 3000G X-Series oscilloscopes, lines connecting acquisition points was always on. Because it is also always on in the HD3-Series oscilloscopes, this command is not necessary.

Discontinued Command	Current Command Equivalent	Comments
:DVM:FREQuency?	:COUNter<c>:CURRent? (see page 387)	The :DVM:FREQuency? query has been replaced by the new Chapter 14 , “:COUNter<c> Commands,” starting on page 385.
:ERASe	:DISplay:CLEar (see page 420)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:EXTernal:PMODE	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:FFT:DMODE	none	FFT display mode options are not available in the HD3-Series oscilloscopes. However, You can enable an FFT Max Function in the user interface.
:FUNction:GOFT:OPERation	:FUNction1:OPERation (see page 561)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:GOFT:SOURce1	:FUNction1:SOURce1 (see page 569)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:GOFT:SOURce2	:FUNction1:SOURce2 (see page 571)	GOFT maps to FUNction1. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:SOURce	:FUNction:SOURce1 (see page 569)	Obsolete command has ADD, SUBTract, and MULTiPLY parameters; current command has GOFT parameter. This was an obsolete command in the 3000G X-Series oscilloscopes.
:FUNction:VIEW	:FUNction:DISPlay (see page 533)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:DESTination	:RECall:FILeName (see page 946) :SAVE:FILeName (see page 946)	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:HARDcopy:FILENAME	:RECall:FILENAME (see page 946) :SAVE:FILENAME (see page 946)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:GRAYscale	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:IGColors	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HARDcopy:PDRiver	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:HWEenable	:STATus:OPERation:HARDware:ENABLE (see page 1325)	Status registers now use SCPI standard commands and queries for controlling and reading the registers.
:HWERegister:CONDition?	:STATus:OPERation:HARDware:CONDition? (see page 1324)	
:HWERegister[:EVENT]?	:STATus:OPERation:HARDware[:EVENT]? (see page 1328)	
:MEASure:ALL	:MEASure:QUICK (see page 723)	There is no "Snapshot All" feature in the HD3-Series oscilloscopes, but there is a "Quick Measure" that will install 10 common measurements on an analog input channel source.
:MEASure:DEFine	:MEASure:DELay:DEFine (see page 685) :MEASure:THResholds:ABSolute (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:DEFine functionality is provided by the current command equivalents.

Discontinued Command	Current Command Equivalent	Comments
:MEASure:LOWer	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TDELta	:MARKer:XDELta (see page 642)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:THResholds	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TMAX	:MEASure:XMAX (see page 766)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TMIN	:MEASure:XMIN (see page 767)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TSTart	:MARKer:X1Position (see page 637)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TSTOp	:MARKer:X2Position (see page 640)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:TVOLt	:MEASure:TVALue (see page 754)	TVALue measures additional values such as db, Vs, etc. This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:MEASure:UPPer	:MEASure:THResholds:ABSolut e (see page 746) :MEASure:THResholds:METHo d (see page 747) :MEASure:THResholds:PERCen t (see page 748) :MEASure:THResholds:STANda rd (see page 749)	The :MEASure:THResholds commands can define absolute values or percentage. This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VDELta	:MARKer:YDELta (see page 649)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VSTArt	:MARKer:Y1Position (see page 646)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VSTOp	:MARKer:Y2Position (see page 648)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MEASure:VTIME	:MEASure:YATX (see page 768)	With :MEASure:YATX, there is a command for installing the measurement on the oscilloscope's display. :MEASure:VTIME had no command, only a query.
:MTEenable	:STATus:OPERation:MTEST:ENA Ble (see page 1351)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:MTERegister[:EVENT]?	:STATus:OPERation:MTEST[:EVE Nt]? (see page 1354)	
:MTEST:AMASK:{SAVE STORE}	:SAVE:MASK[:START] (see page 963)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:AVERage	:ACQuire:TYPE AVERage (see page 311)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:AVERage:COUNT	:ACQuire:COUNT (see page 288)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:LOAD	:RECall:MASK[:START] (see page 948)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTEST:RUMode	:MTEST:RMODE (see page 813)	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:MTESt:RUMode:SOFailure	:MTESt:RMODE:FACTion:STOP (see page 817)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTESt:{START STOP}	:RUN (see page 275) or :STOP (see page 279)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:MTESt:TRIGger:SOURce	:TRIGger Commands (see page 1479)	There are various commands for setting the source with different types of triggers. This was an obsolete command in the 3000G X-Series oscilloscopes.
:OPEE	:STATus:OPERation:ENABLE (see page 1299)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:OPERRegister:CONDition?	:STATus:OPERation:CONDition? (see page 1298)	
:OPERRegister[:EVENT]?	:STATus:OPERation[:EVENT]? (see page 1302)	
:OVLenable	:STATus:OPERation:OVERload:ENABLE (see page 1364)	Status registers now use SCPI standard commands and queries for controlling and reading status registers.
:OVLRegister?	:STATus:OPERation:OVERload[:EVENT]? (see page 1367)	
:PRINT?	:DISPlay:DATA? (see page 425)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SAVE:IMAGe:AREA	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SAVE:IMAGe:PALette	none	Color images are always saved in the HD3-Series oscilloscopes.
:SBUS<n>:CAN:FDSTandard	none	This is no longer an option. The ISO standard is used when decoding or triggering on FD frames.
:SBUS<n>:LIN:SIGNAL:DEFinitio n	none	This was an obsolete command in the 3000G X-Series oscilloscopes.
:SYSTem:MENU	none	This was an obsolete command in the 3000G X-Series oscilloscopes.

Discontinued Command	Current Command Equivalent	Comments
:SYSTem:PRECision	:SYSTem:PRECision:LENGth (see page 1443)	Precision analysis is no longer a feature to be enabled. Instead, you simply choose the desired analysis record length.
:TRIGger:DElay:TRIGger:COUNt	none	You cannot trigger on an Nth edge.
:TRIGger:THReshold	:POD<n>:THReshold (see page 835) or :DIGital<d>:THReshold (see page 404)	This was an obsolete command in the 3000G X-Series oscilloscopes.
:TRIGger:ZONE:STATe	none	There is no longer an overall zone trigger feature on/off state.
:WAVEform:VIEW	none	In the HD3-Series oscilloscopes, all captured data is used.

Discontinued Parameters Some previous oscilloscope queries returned control setting values of OFF and ON. The InfiniiVision HD3-Series oscilloscopes return only the enumerated values 0 (for off) and 1 (for on).

:CHANnel<n>:BWLimit

O (see [page 1708](#))

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 20 MHz.

NOTE

This command is provided to support programs from earlier legacy InfiniiVision oscilloscopes. The HD3-Series oscilloscope command for limiting the bandwidth and increasing vertical resolution is :CHANnel<n>:BANDwidth.

Query Syntax :CHANnel<n>:BWLimit?

The :CHANnel<n>:BWLimit? query returns the current setting of the channel bandwidth limit.

Return Format <bwlimit><NL>

<bwlimit> ::= {1 | 0}

See Also • [":CHANnel<n>:BANDwidth"](#) on page 349

:MEASure:SCRatch

O (see [page 1708](#))

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 676](#)) instead.

:SBUS<n>:SPI:SOURce:DATA

O (see [page 1708](#))

Command Syntax :SBUS<n>:SPI:SOURce:DATA <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>: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 1479
 - [":SBUS<n>:SPI:SOURce:MOSI"](#) on page 1112
 - [":SBUS<n>:SPI:SOURce:MISO"](#) on page 1111
 - [":SBUS<n>:SPI:SOURce:CLOCK"](#) on page 1109
 - [":SBUS<n>:SPI:SOURce:FRAME"](#) on page 1110
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:DATA"](#) on page 1113
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:DATA"](#) on page 1115
 - [":SBUS<n>:SPI:TRIGger:PATtern:MISO:WIDTH"](#) on page 1114
 - [":SBUS<n>:SPI:TRIGger:PATtern:MOSI:WIDTH"](#) on page 1116

:TIMEbase[:MAIN]:DELay

O (see [page 1708](#))

Command Syntax :TIMEbase[:MAIN]: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[:MAIN]: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 1473](#)).

NOTE

The :TIMEbase[:MAIN]:DELay command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :TIMEbase[:MAIN]:POSition command (see [page 1469](#)) instead.

Query Syntax :TIMEbase[:MAIN]:DELay?

The :TIMEbase[:MAIN]: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 49](#), "Programming Examples," starting on page 1717

:TRIGger:DElay:TDElay:TIME

O (see [page 1708](#))

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.

NOTE

This command is now mapped to **:TRIGger:DElay:TDElay:TMIN** on page 1502.

NOTE

Entered delay times automatically adjust to the nearest valid increment of ~156.5 ps.

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 1479
 - **:TRIGger:DElay:TRIGger:SLOPe** on page 1503

45 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.

-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

46 Status Reporting

Status Reporting Data Structures / 1672

Output Queue / 1673

Message Queue / 1674

Clearing Registers and Queues / 1675

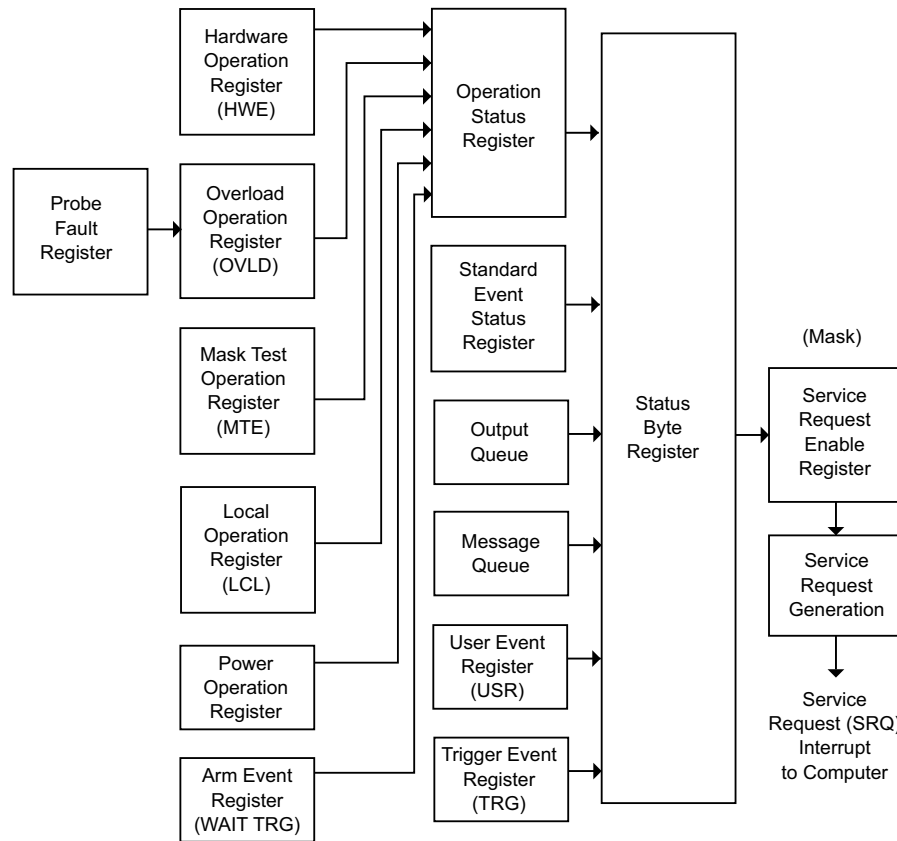
Status Reporting Decision Chart / 1676

Example: Checking for Armed Status / 1677

Example: Waiting for IO Operation Complete / 1682

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 Register and the Overload Operation 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 status registers, the summary bit is set if an event in its Condition register, as defined by its Transition Filters, is latched into the Event register and the corresponding bit in its Enable register is set (see **"Introduction to :STATus Commands"** on page 1283). Events latched to an Event register remain set until the register is queried or cleared. Registers are queried 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.

- See Also
- **"Service Request Enable Register"** on page 251
 - **"Status Byte Register"** on page 253
 - **"Trigger Event Register"** on page 1395
 - **"User Event Register"** on page 1408
 - **"Message Queue"** on page 1674
 - **"Output Queue"** on page 1673
 - **"Standard Event Status Register"** on page 239
 - **"Operation Status Register"** on page 1289
 - **"Arm Event Register"** on page 1304
 - **"Power Operation Register"** on page 1382
 - **"Local Operation Register"** on page 1330
 - **"Mask Test Operation Register"** on page 1343
 - **"Overload Operation Register"** on page 1356
 - **"Probe Fault Register"** on page 1369
 - **"Hardware Operation Register"** on page 1317

Status Reporting Data Structures

The status register bits are described in the following tables:

- **Table 87** – Summary message goes to Status Byte Register bit 6.
- **Table 88**
 - **Table 162** – Summary message goes to Status Byte Register bit 0 (TRG).
 - **Table 164** – Summary message goes to Status Byte Register bit 1 (USR).
 - **Table 86** – Summary message goes to Status Byte Register bit 5 (ESB).
 - **Table 146** – Summary message goes to Status Byte Register bit 7 (OPER).
 - **Table 148** – Summary message goes to Operation Status Register bit 5 (WAIT TRIG).
 - **Table 160** – Summary message goes to Operation Status Register bit 7 (Power).
 - **Table 152** – Summary message goes to Operation Status Register bit 8 (Lcl).
 - **Table 154** – Summary message goes to Operation Status Register bit 9 (MTE).
 - **Table 156** – Summary message goes to Operation Status Register bit 11 (OVLD).
 - **Table 158** – Summary message goes to Overload Operation Register bit 12 (Channel Probe Fault Summary)
 - **Table 150** – Summary message goes to Operation Status Register bit 12 (HWE).

The status registers hierarchy 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.

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

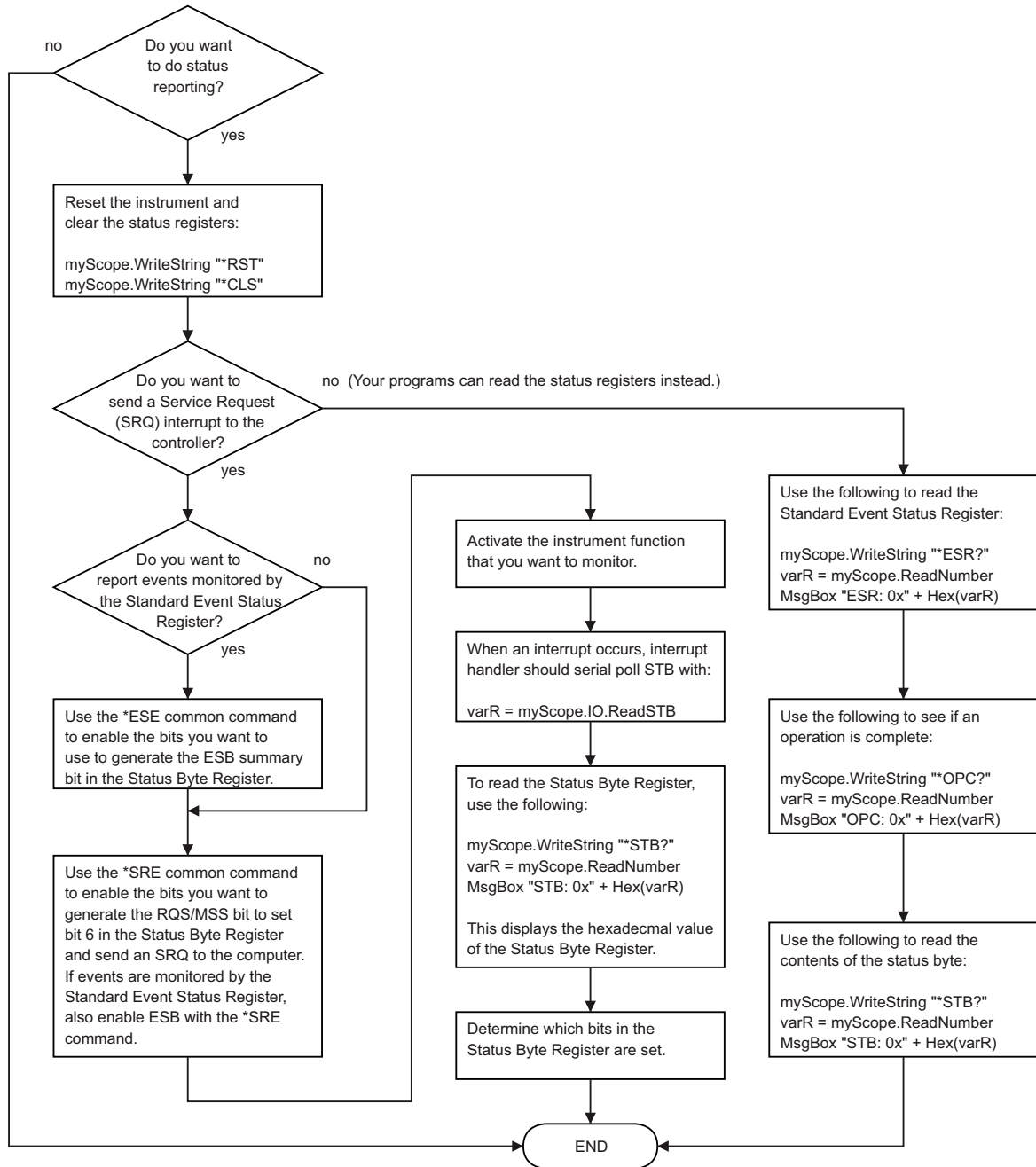
The message queue contains the text of the last message written to the advisory line on the screen of the oscilloscope. The length of the oscilloscope's message queue is 1. Note that messages sent with the :SYSTem:DSP command do not set the MSG status bit in the Status Byte Register.

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.

See Also • **"*CLS (Clear Status)"** on page 236

Status Reporting Decision Chart



Example: Checking for Armed Status

```

#!/python3
# -*- coding: utf-8 -*-

# *****
# This script using the Python language (http://www.python.org/) and
# the PyVISA package (http://pyvisa.readthedocs.org/) shows three
# methods to tell whether a Keysight InfiniiVision oscilloscope is
# armed.
# *****

# Import modules
# -----
import sys
import pyvisa
import time

# Initialization constants
# -----
# Get VISA address from Keysight IO Libraries Connection Expert
VISA_ADDRESS = "USB0::0x2A8D::0x4704::Unset::0::INSTR"
GLOBAL_TOUT = 20000 # IO timeout in milliseconds

# =====
# Method 1: Query the Armed Event Register with :AER?
# -----
# This method reads the 1-bit Armed Event Register using the :AER?
# query.
#
# The Armed Event Register bit goes low (0) when it is read using
# :AER? or when a *CLS command is issued.
# =====
def method_1():

    # Stop the oscilloscope.
    InfiniiVisionHD.query(":STOP;*OPC?")

    # Method 1: Initiate capture using :SINGLe
    # -----
    print("Acquiring signal (Method 1, using :SINGLe)...\n")
    now = time.perf_counter()

    # Clear all status registers before checking for new events.
    InfiniiVisionHD.write("*CLS")

    # Because the :AER? query will not work with :DIGitize (which is
    # blocking), use the :SINGLe command to start the acquisition.
    InfiniiVisionHD.write(":SINGLe")

    # Method 1: Determine if armed using :AER? query.
    # -----
    # Define armed criteria.
    ARMED = 1

```

```

# Test for armed.
ARMED_STATUS = int(InfiniiVisionHD.query(":AER?"))

# Wait indefinitely until armed.
while ARMED_STATUS != ARMED:
    # Check the status again after small delay.
    time.sleep(0.1)    # 100 ms delay to prevent excessive queries.
    ARMED_STATUS = int(InfiniiVisionHD.query(":AER?"))

print("Oscilloscope is armed (method 1, using :AER? query)!")
print("It took " + str(time.perf_counter() - now) + \
      " seconds to arm.\n")

# =====
# Method 2: Read the Status Byte
# -----
# This method reads the Status Byte register's OPER bit (bit 7) using
# the "read status byte" function in VISA, which works during blocking
# commands and can therefore be used with the :DIGitize command.
#
# The Status Byte bits do NOT go low (0) when the register is read.
#
# The *CLS command will clear the Status Byte bits.
#
# CAUTION: The oscilloscope's status registers are not updated until
# the :DIGitize completes. So, while the ARM may go true midway
# through the :DIGitize, it does not get reported to the status model
# until the :DIGitize completes, and therefore the STB does not
# reflect it as true until the :DIGitize completes.
# =====
def method_2():

    # Stop the oscilloscope.
    InfiniiVisionHD.query(":STOP;*OPC?")

    # Method 2: Initiate capture using :DIGitize or :SINGLe
    # -----
    print("Acquiring signal (Method 2, using :DIGitize)...\n")
    now = time.perf_counter()

    # Clear all status registers before checking for new events.
    InfiniiVisionHD.write("*CLS")

    # Mask out all bits in the Operation Status Event Register except
    # for the ARM bit. "Unmask" only the arm bit.
    InfiniiVisionHD.write(":STATus:OPERation:ENABle 32")

    # Use the :DIGitize command to start the acquisition.
    InfiniiVisionHD.write(":DIGitize")

    # Method 2: Determine if armed by reading the Status Byte.
    # -----
    # Define register bit masks for the Status Byte Register
    ARM_BIT = 7
    # 1 leftshift 7 = 128 (bit 7 in the Status Byte Register)

```

```

ARM_MASK = 1 << ARM_BIT

# Define armed criteria.
ARMED = 1 << ARM_BIT    # 1 leftshift 7 = 128

# Test for armed.
STATUS_BYTE = int(InfiniiVisionHD.read_stb())
ARMED_STATUS = STATUS_BYTE & ARM_MASK
# Note that you could also do:
# ARMED_STATUS = int(InfiniiVisionHD.query("*STB?"))
# BUT *STB? does not work with the blocking :DIGitize.

# Wait indefinitely until armed.
while ARMED_STATUS != ARMED:
    # Check the status again after small delay.
    time.sleep(0.1)    # 100 ms delay to prevent excessive queries.
    STATUS_BYTE = int(InfiniiVisionHD.read_stb())
    ARMED_STATUS = STATUS_BYTE & ARM_MASK

print("Oscilloscope is armed (method 2, using Read STB function)!")
print("It took " + str(time.perf_counter() - now) + \
      " seconds to arm.\n")

# =====
# Method 3: Query the Operation Status Event Register with
# :STATus:OPERation:EVENT?
# -----
# This method reads the Operation Status Event Register's Wait Trig
# bit (bit 5) using the :STATus:OPERation:EVENT? query.
#
# The Operation Status Event Register bits are cleared (0) when the
# register is read.
#
# Also, the Wait Trig bit does NOT go low (0) when the oscilloscope
# becomes unarmed by starting or stopping another acquisition (before
# the first one finishes) or by changing the time scale.
#
# The Wait Trig bit is cleared by a *CLS command, by reading the
# Operation Status Event Register with the :STATus:OPERation:EVENT?
# query, or by reading the Armed Event Register register with the
# :AER? query.
# =====
def method_3():

    # Stop the oscilloscope.
    InfiniiVisionHD.query(":STOP;*OPC?")

    # Method 3: Initiate capture using :SINGle
    # -----
    print("Acquiring signal (Method 3, using :SINGle)...\n")
    now = time.perf_counter()

    # Clear all status registers before checking for new events.
    InfiniiVisionHD.write("*CLS")

    # Because the :STATus:OPERation:EVENT? query will not work with

```

```

# :DIGitize (which is blocking), use the :SINGle command to
# start the acquisition.
InfiniiVisionHD.write(":SINGle")

# Method 3: Determine if armed using :STATus:OPERation:EVENT?
# query.
# -----
# Define bit masks for the Operation Status Event Register
ARM_BIT = 5
# 1 leftshift 5 = 32 (bit 5 in the Operation Status Event
# Register)
ARM_MASK = 1 << ARM_BIT

# Define armed criteria.
ARMED = 1 << ARM_BIT # 1 leftshift 5 = 32

# Test for armed.
STATUS_REGISTER = \
    int(InfiniiVisionHD.query(":STATus:OPERation:EVENT?"))
ARMED_STATUS = STATUS_REGISTER & ARM_MASK

# Wait indefinitely until armed.
while ARMED_STATUS != ARMED:
    # Check the status again after small delay.
    time.sleep(0.1) # 100 ms delay to prevent excessive queries.
    STATUS_REGISTER = \
        int(InfiniiVisionHD.query(":STATus:OPERation:EVENT?"))
    ARMED_STATUS = STATUS_REGISTER & ARM_MASK

print("Oscilloscope is armed (method 3, using " + \
      ":STATus:OPERation:EVENT? query)!")
print("It took " + str(time.perf_counter() - now) + \
      " seconds to arm.\n")

# =====
# Main
# =====

# Connect and initialize oscilloscope
# -----
# Define VISA Resource Manager & Install directory
rm = pyvisa.ResourceManager('C:\\Windows\\System32\\visa64.dll')

# Define and open the oscilloscope using the VISA address
InfiniiVisionHD = rm.open_resource(VISA_ADDRESS)

# Set the Global Timeout
InfiniiVisionHD.timeout = GLOBAL_TOUT

# Clear the instrument bus
InfiniiVisionHD.clear()

# Reset the oscilloscope.
InfiniiVisionHD.write("*RST")

# Autoscale to set up vertical scale and trigger level on Probe Comp.

```

```

InfiniiVisionHD.write(":CHANnel1:DISPlay OFF")
InfiniiVisionHD.write(":CHANnel2:DISPlay ON")
InfiniiVisionHD.write(":AUToscale:CHANnels DISplayed")
InfiniiVisionHD.write(":AUToscale")

# Ensure a "long" time to arm (5 seconds) and not trigger immediately.
# -----
# 10 second total capture, with trigger point in the middle = 5s to arm
InfiniiVisionHD.write(":TIMEbase:RANGe 10")
# Prevent Auto trigger.
InfiniiVisionHD.write(":TRIGger:SWEep NORMal")

# Use the three methods to check whether the oscilloscope is armed.
# -----
method_1()
method_2()
method_3()

# End of Script
# -----
InfiniiVisionHD.clear()    # Clear communications interface
InfiniiVisionHD.close()   # Close communications interface
print("All done.")

```

Example: Waiting for IO Operation Complete

NOTE

The IOC (IO Operation Complete) and IOF (IO Operation Failed) bits in the Operation Status Event Register identify when :WMEMory<r>:SAVE and other :SAVE and :RECall commands are completely done. To determine when a IO operation is complete, you should use these bits instead of the OPC (Operation Complete) bit in the Standard Event Status Register or the *OPC? query.

```
#!/python3
# -*- coding: utf-8 -*-

# *****
# This script using the Python language (http://www.python.org/) and
# the PyVISA package (http://pyvisa.readthedocs.org/) shows how to
# wait for IO operation completion in a Keysight InfiniiVision
# oscilloscope.
# *****

# Import modules
# -----
import sys
import pyvisa
import time

# Initialization constants
# -----
# Get VISA address from Keysight IO Libraries Connection Expert
VISA_ADDRESS = "USB0::0x2A8D::0x4704::Unset::0::INSTR"
GLOBAL_TOUT = 20000 # IO timeout in milliseconds

# =====
# Check for IO Operation Complete using :STATus:OPERation:EVENT? query.
# -----
# This method reads the Operation Status Register's IOC bit
# (bit 13) using the :STATus:OPERation:EVENT? query.
#
# The Operation Status Event Register bits are cleared (0) when the
# register is read.
#
# All status bits are cleared by a *CLS command.
# =====
def wait_io_operation():

    # -----
    now = time.perf_counter()
    # Define bit masks for the Operation Status Event Register
    IOC_BIT = 13
    # 1 leftshift 13 = 8192 (bit 13 in the Operation Status Event
    # Register)
    IOC_MASK = 1 << IOC_BIT

    # Define IO complete criteria.
```

```

IO_COMPLETE = 1 << IOC_BIT    # 1 leftshift 13 = 8192

# Test for armed.
STATUS_REGISTER = \
    int(InfiniiVisionHD.query(":STATUS:OPERation:EVENT?"))
IO_COMPLETE_STATUS = STATUS_REGISTER & IOC_MASK

# Wait indefinitely until armed.
while IO_COMPLETE_STATUS != IO_COMPLETE:
    # Check the status again after small delay.
    time.sleep(0.1)    # 100 ms delay to prevent excessive queries.
    STATUS_REGISTER = \
        int(InfiniiVisionHD.query(":STATUS:OPERation:EVENT?"))
    IO_COMPLETE_STATUS = STATUS_REGISTER & IOC_MASK

time_in_seconds = str(time.perf_counter() - now)
print("IO Operation Complete (from :STATUS:OPERation:EVENT? query).")
)

print("It took %s seconds for IO operation to complete.\n" % \
      time_in_seconds)

# =====
# Main
# =====

# Connect and initialize oscilloscope
# -----
# Define VISA Resource Manager & Install directory
rm = pyvisa.ResourceManager('C:\\Windows\\System32\\visa64.dll')

# Define and open the oscilloscope using the VISA address
InfiniiVisionHD = rm.open_resource(VISA_ADDRESS)

# Set the Global Timeout
InfiniiVisionHD.timeout = GLOBAL_TOUT

# Clear the instrument bus
InfiniiVisionHD.clear()

# Reset the oscilloscope.
# InfiniiVisionHD.write("*RST")
# Or comment out to use the current oscilloscope setup.

# Autoscale to set up vertical scale and trigger level on channels 1 and
# 2.
InfiniiVisionHD.write(":CHANnel1:DISPlay ON")
InfiniiVisionHD.write(":CHANnel2:DISPlay ON")
InfiniiVisionHD.write(":AUToscale:CHANnels DISPlayed")
InfiniiVisionHD.write(":AUToscale")

# Between :WMEemory<r>:SAVE commands, wait for IO complete.
# -----
# Clear all status registers before checking for new events.
InfiniiVisionHD.write("*CLS")

InfiniiVisionHD.write(":WMEemory1:SAVE CHANnel1")

```

```
wait_io_operation()
InfiniiVisionHD.write(":WMEemory2:SAVE CHANnel2")
wait_io_operation()

# End of Script
# -----
InfiniiVisionHD.clear()    # Clear communications interface
InfiniiVisionHD.close()    # Close communications interface
print("All done.")
```


47 Synchronizing Acquisitions

Synchronization in the Programming Flow / 1686

Blocking Synchronization / 1687

Polling Synchronization With Timeout / 1688

Synchronizing with a Single-Shot Device Under Test (DUT) / 1690

Synchronization with an Averaging Acquisition / 1693

Example: Blocking and Polling Synchronization / 1695

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 1686](#)).
- 2 Acquire a waveform (see [page 1686](#)).
- 3 Retrieve results (see [page 1686](#)).

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 " Blocking Synchronization " on page 1687.	See " Polling Synchronization With Timeout " on page 1688.

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("USB0::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 10000 ' Set I/O communication timeout.

    ' 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

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Lo
ng)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 60000 ' Set I/O communication timeout.

    ' 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"
    myScope.WriteString ":TRIGger:SWEep NORMAl"

    ' Stop acquisitions and wait for the operation to complete.
    myScope.WriteString ":STOP"
    myScope.WriteString "*OPC?"
    strQueryResult = myScope.ReadString
    myScope.WriteString ":DISPlay:CLEar"

    ' 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 = 20000 ' 20 seconds.
lngElapsed = 0

Do While lngElapsed <= lngTimeout
    myScope.WriteString ":STATUS:OPERation: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 1687 and **"Polling Synchronization With Timeout"** on page 1688 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 1688 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

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If

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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear ' Clear the interface.
    myScope.IO.Timeout = 60000 ' Set I/O communication timeout.

    ' 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"
myScope.WriteString ":TRIGger:SWEep NORMal"

' Stop acquisitions and wait for the operation to complete.
myScope.WriteString ":STOP"
myScope.WriteString "*OPC?"
strQueryResult = myScope.ReadString
myScope.WriteString ":DISPlay:CLear"

' 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 = 20000 ' 20 seconds.
lngElapsed = 0

Do While lngElapsed <= lngTimeout
    myScope.WriteString ":STATus:OPERation: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

#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMillisecon
ds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Lo
ng)
#End If

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::0x2A8D::0x4704::Unset::0::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

```

Example: Blocking and Polling Synchronization

```

#!/python3
# -*- coding: utf-8 -*-

# *****
# This script using the Python language (http://www.python.org/) and
# the PyVISA package (http://pyvisa.readthedocs.org/) shows the two
# best synchronization methods for InfiniiVision oscilloscopes.
# Benefits and drawbacks of each method are described. No error
# handling is provided except in the actual synchronization methods.
# *****

# Import modules
# -----
import sys
import pyvisa
import time

# Initialization constants
# -----
# Get VISA address from Keysight IO Libraries Connection Expert
VISA_ADDRESS = \
    "USB0::0x2A8D::0x4704::Unset::0::INSTR"
GLOBAL_TOUT = 10000 # IO timeout in milliseconds

TIME_TO_TRIGGER = 10 # Time in seconds
# -----
# This is the time until the FIRST trigger event.
#
# While the script calculates a general timeout for the given setup,
# it cannot know when a trigger event will occur. Thus, you must
# still set this value.
#
# This time is in addition to the calculated minimum timeout... so, if
# an oscilloscope might take say, 1 us to arm and acquire data, the
# signal might take 100 seconds before it occurs... this accounts for
# that.
#
# The SCOPE_ACQUISITION_TIME_OUT calculation pads this by 1.1.
# -----

TIME_BETWEEN_TRIGGERS = 0.025 # Time in seconds - for Average,
# Segmented, and Equivalent Time modes, else set to 0
# -----
# In Average, Segmented, and Equivalent Time modes, the oscilloscope
# makes repeated acquisitions. This is similar to the above
# TIME_TO_TRIGGER, but it is the time BETWEEN triggers. For example,
# it might take 10 seconds for the first trigger event, and then they
# might start occurring regularly at say, 1 ms intervals. In that
# scenario, 15 seconds (a conservative number for 10s) would be good
# for TIME_TO_TRIGGER, and 2 ms (again conservative) would be good for
# TIME_BETWEEN_TRIGGERS.
#
# The default in this sample script is 0.025 seconds. This is to make

```

```

# the sample work for the LINE trigger used in this script when the
# oscilloscope is in Average, Segmented, and Equivalent Time modes to
# force a trigger off of the AC input line (:TRIGger:EDGE:SOURce LINE)
# which runs at 50 or 60 Hz in most of the world (1/50 Hz -> 20 ms, so
# use 25 ms to be conservative).
#
# The SCOPE_ACQUISITION_TIME_OUT calculation pads this by 1.1.
# -----

# =====
# Define a simple and fast function utilizing the blocking :DIGitize
# command in conjunction with *OPC?.
# -----
#
# Benefits of this method:
#
# - Fastest, compact
# - Only way for Average Acquisition type:
#   - The :SINGLE command does not do a complete average.
#   - Counting triggers with :RUN is much too slow.
# - Allows for easy synchronization with math functions
# - Don't have to deal with the status registers, which can be
#   confusing.
# - Potentially faster than polling_method(), for better throughput
# - Because it's faster, one can retrieve more accurate acquisition
#   times than with a polling method.
# - Works best for segmented memory if any post processing is done
#   on the oscilloscope, for example, measurements, lister, math,
#   as this does not come back until the processing is all done
#   - In this scenario, :DIGitize does not reduce the sample
#     rate or memory depth.
#
# Drawbacks of this method:
#
# - Usually does not fill acquisition memory to the maximum
#   available, usually only on-screen data.
#
# - May not be at the highest sample rate (compared with the
#   polling_method).
#
# - Requires a well-chosen, hard-set timeout that will cover the
#   time to arm, trigger, and finish acquisition.
#
# - Requires Exception handling and a device_clear() for a
#   possible timeout (no trigger event).
#
# - Socket connection cannot do device_clear()
#
# - Because :DIGitize is a "specialized form of the :RUN" command,
#   on these oscilloscopes, that results in:
#
#   - the sample rate MAY be reduced from using :SINGLE -
#     usually at longer time scales - typically only acquires
#     what is on screen, though at the fastest time scales,
#     more than on screen data may be acquired. Thus, for max
#     memory and max sample rate, use the polling_method(),

```

```

#           which uses :SINGLE.
#
# How it works:
#
# - The :DIGitize command is a blocking command, and thus, all
#   other SCPI commands are blocked until :DIGitize is completely
#   done. This includes any subsequent processing that is already
#   set up, such as math and measurements.
#
# KEY POINT: However, :DIGitize does not prevent additional
# commands from being sent to the queue or cause the remote
# program to wait. For example, if your program does something
# like:
#
#     InfiniiVisionHD.write(":DIGitize")
#     print("Signal acquired.\n")
#
# The "Signal acquired" message will be written immediately
# after the :DIGitize is sent, not after the acquisition and
# processing is complete.
#
# To pause the program until the :DIGitize is complete, you must
# wait for a query result after the :DIGitize. For example, in
# this case:
#
#     query_result = InfiniiVisionHD.query(":DIGitize;*OPC?")
#     print("Signal acquired.\n")
#
# The "Signal acquired" message will be written after the
# acquisition and processing is complete. The *OPC? query is
# appended to :DIGitize with a semi-colon (;), which
# essentially ties it to the same thread in the parser. It is
# immediately dealt with once :DIGitize finishes and gives a "1"
# back to the program (whether the program uses it or not),
# allowing the program to move on.
#
# Other Notes:
#
# - If you DO NOT know when a trigger will occur, you should set a
#   very long timeout.
#
# - The timeout should be adjusted before and after the :DIGitize
#   operation, though this is not absolutely required.
#
# - A :DIGitize can be aborted with a device clear, which also
#   stops the oscilloscope:
#   InfiniiVisionHD.clear()
#
# - :DIGitize disables the anti-aliasing feature (sample rate
#   dither) on all InfiniiVision and InfiniiVision X-Series
#   oscilloscopes.
#
# - :DIGitize temporarily blocks the front panel, and all front
#   panel presses are queued until :DIGitize is done. So if you
#   change the vertical scale, it will not happen until the
#   acquisition is done.
#
#

```

```

# The exception is that the Run/Stop button on the front panel
# is NOT blocked (unless the front panel is otherwise locked by
# ":SYSTem:LOCK ON").
# =====
def blocking_method():

    InfiniiVisionHD.timeout = SCOPE_ACQUISITION_TIME_OUT
    # Time in milliseconds (PyVISA uses ms) to wait for the
    # oscilloscope to arm, trigger, finish acquisition, and finish
    # any processing.
    #
    # Note that this is a property of the device interface,
    # InfiniiVisionHD.
    #
    # If doing repeated acquisitions, this should be done BEFORE the
    # loop, and changed again after the loop if the goal is to
    # achieve the best throughput.

    print("Acquiring signal(s)...")
    # Set up a try/except block to catch a possible timeout and exit.
    try:
        InfiniiVisionHD.query(":DIGitize;*OPC?")
        # Acquire the signal(s) with :DIGitize (blocking) and wait
        # until *OPC? comes back with a one.
        print("Signal acquired.")

        # Reset timeout back to what it was, GLOBAL_TOUT.
        InfiniiVisionHD.timeout = GLOBAL_TOUT

    # Catch a possible timeout and exit.
    except Exception:
        print("The acquisition timed out, most likely due to no " \
              "trigger or improper setup causing no trigger. " \
              "Properly closing the oscilloscope connection and " \
              "exiting script.\n")
        InfiniiVisionHD.clear() # Clear communications interface;
                                # A device clear also aborts digitize.
        InfiniiVisionHD.close() # Close communications interface
        sys.exit("Exiting script.")

# =====
# Define a function using the non-blocking :SINGle command and polling
# on the Operation Status Condition Register
# -----
#
# Benefits of this method:
#
# - Don't have to worry about interface timeouts.
# - Easy to expand to know when the oscilloscope is armed.
# - Don't have to worry about interface timeouts
# - Easy to expand to know when scope is armed, and triggered
# - MAY result in a higher sample rate than the blocking method
# - Always fills max available memory
# - Can use with a socket connection if desired
#
#

```

```

# Drawbacks of this method:
#
# - Slow, as you don't want to poll the oscilloscope too fast.
#
# - DOES NOT work for Equivalent time mode. MUST use the blocking
#   method.
#
# - Slow
#
# - Does NOT work for Average Acquisition type
#   - :SINGLE does not do a complete average
#     - It does a single acquisition as if it were in NORMAL
#       acq. type
#     - Counting triggers in :RUN is much too slow
#
# - Works for Segmented Memory, BUT if any post processing is done
#   on the oscilloscope, for example measurements, lister, math, as
#   this reports that the acquisition is done, which is correct,
#   BUT the processing is NOT done, and it will take an indefinite
#   amount of time to wait for that, though there is no way to tell
#   if it is done. Use the blocking_method for Segmented Memory.
#
# - Can't be used effectively for synchronizing math functions
#
#   It can be done by applying an additional hard coded wait after
#   the acquisition is done. At least 200 ms is suggested, more
#   may be required.
#
#   However, as long as the timeout is not excessively short, the
#   math happens fast enough that once :STATus:OPERation:CONDition?
#   comes back as done that one can just wait for it when it is
#   time to pull the math waveform.
#
# - Still need some maximum timeout (here MAX_TIME_TO_WAIT),
#   ideally, or the script will sit in the while loop forever if
#   there is no trigger event
#
#   Max timeout (here MAX_TIME_TO_WAIT) must also account for any
#   processing done (see comments on math above)
#
#   Max timeout (here MAX_TIME_TO_WAIT) must also account for time
#   to arm the scope and finish the acquisition
#
#   This arm/trigger/finish part is accounted for in the main script.
#
# How it works:
#
# - What really matters is the RUN bit in the Operation Condition
#   (not Event) Register. This bit changes based on the
#   oscilloscope state.
#
#   If the oscilloscope is running, it is high (8), and low (0) if
#   it is stopped.
#
#   The only (best) way to get at this bit is with the
#   :STATus:OPERation:CONDition? query. The Operation Condition Regis
#   ter

```

```

# can reflect states for other oscilloscope properties, for
# example, if the oscilloscope is armed, thus it can produce
# values other than 0 (stopped) or 8 (running).
#
# To handle that, the result of :STATus:OPERation:CONDition? is bitwise
# ANDed (& in Python) with an 8. This is called "unmasking".
#
# Here, the "unmasking" is done in the script. On the other
# hand, it is possible to "mask" which bits get passed to the
# summary bit to the next register below on the instrument
# itself. However, this method is typically only used when
# working with the Status Byte, and not used here.
#
# - Why 8 = running = not done?
#
# The Run bit is the 4th bit of the Operation Status Condition
# (and Event) Registers.
#
# The registers are binary and start counting at zero, thus the
# 4th bit is bit number 3, and  $2^3 = 8$ , and thus it returns an
# 8 for high and a 0 for low.
#
# - Why the CONDITION and NOT the EVENT register?
#
# The Condition register reflects the CURRENT state, while the
# EVENT register reflects the first event that occurred since it
# was cleared or read (as in: has it EVER happened?), thus the
# CONDITION register is used.
#
# Note that with this method using :SINGle, for InfiniiVision
# X-Series oscilloscopes only, :SINGle itself forces the trigger
# sweep mode into NORMal. This does not happen with the blocking
# method, using :DIGitize, or on the InfiniiVision notXs.
# =====
def polling_method():

    MAX_TIME_TO_WAIT = SCOPE_ACQUISITION_TIME_OUT/float(1000)
    # Time in seconds to wait for the oscilloscope to arm, trigger,
    # and finish acquisition.
    #
    # Note that this is NOT a property of the device interface,
    # InfiniiVisionHD, but rather some constant in the script to be
    # used later with the Python module "time", and will be used with
    # time.perf_counter().

    # Define "mask" bits.
    #
    # Mask condition for Run state in the Operation Status Condition
    # (and Event) Register.
    #
    # Refer to Programmer's Guide chapters on Status Reporting and
    # Synchronizing Acquisitions.
    RUN_BIT = 3
    # The run bit is the 4th bit (see next set of comments @
    # Completion Criteria).
    RUN_MASK = 1<<RUN_BIT

```



```

# This basically means: 2^3 = 8, or rather, in Python 2**3
# (<< is a left shift; left shift is fastest); this is used later
# to "unmask" the result of the Operation Status Event Register
# as there is no direct access to the RUN bit.

# Define completion criterion:
ACQ_DONE = 0
# Means the oscilloscope is stopped
ACQ_NOT_DONE = 1<<RUN_BIT
# Means the oscilloscope is running; value is 8. This is the
# 4th bit of the Operation Status Condition (and Event) Register.
# The registers are binary and start counting at zero, thus the
# 4th bit (4th position in a binary representation of decimal
# 8 = 2^3 = (1 left shift 3). This is either High (running = 8)
# or low (stopped and therefore done with acquisition = 0).

print("Acquiring signal(s)...")

# Define acquisition start time. This is in seconds.
StartTime = time.perf_counter()

# Begin Acquisition with *CLS and the non-blocking :SINGLe
# command, concatenated together. The *CLS clears all (non-mask)
# registers & sets them to 0;
InfiniiVisionHD.write("*CLS;:SINGLe")

# Initialize the loop entry condition (assume Acq. is not done).
Acq_State = ACQ_NOT_DONE

# Poll the oscilloscope until Acq_State is a one. (This is NOT a
# "Serial Poll.")
while Acq_State == ACQ_NOT_DONE and (time.perf_counter() - StartTime
                                     <= MAX_TIME_TO_WAIT):

    # Ask oscilloscope if it is done with the acquisition via the
    # Operation Status Condition (not Event) Register.
    # The Condition register reflects the CURRENT state, while
    # the EVENT register reflects the first event that occurred
    # since it was cleared or read, thus the CONDITION register
    # is used.
    #
    # DO NOT do:
    # InfiniiVisionHD.query("*CLS;SINGLe;STATus:OPERation:CONDition?")
    #
    # as putting :STATus:OPERation:CONDition? right after :SINGLe
    # doesn't work reliably
    #
    # The oscilloscope SHOULD trigger, but it sits there with the
    # Single hard key on the oscilloscope lit yellow; pressing
    # this key causes a trigger.
    Status = int(InfiniiVisionHD.query(":STATus:OPERation:CONDition?"))

    # Bitwise AND of the Status and RUN_MASK. This exposes ONLY
    # the 4th bit, which is either High (running = 8) or low
    # (stopped and therefore done with acquisition = 0)
    Acq_State = (Status & RUN_MASK)

```

```

if Acq_State == ACQ_DONE:
    break # Break out of while loop so that the 100 ms pause
        # below is not incurred if done.
time.sleep(0.1) # Pause 100 ms to prevent excessive queries
# This can actually be set a little faster, at 0.045. The
# point here is that:
#
# 1. If there are other things being controlled, going too
#    fast can tie up the bus.
# 2. Going faster does not work on all scopes. The
#    symptom of this not working is:
#
# The oscilloscope SHOULD trigger, but it sits there with the
# Single hard key on the oscilloscope lit yellow; pressing
# this key causes a trigger.
#
# The pause should be at the end of the loop, so that the
# oscilloscope is immediately asked if it is done.
#
# Loop exits when Acq_State != NOT_DONE, that is, it exits
# the loop when it is DONE or if the max wait time is
# exceeded.

if Acq_State == ACQ_DONE: # Acquisition fully completed
    print("Signal acquired.")
else: # Acquisition failed for some reason
    print("Max wait time exceeded.")
    print("This happens if there was no trigger event.")
    print("Adjust settings accordingly.\n")
    print("Properly closing oscilloscope connection and exiting " \
          "script.\n")
    InfiniiVisionHD.clear() # Clear communications interface
    InfiniiVisionHD.query(":STOP;*OPC?")
    # Stop the oscilloscope
    InfiniiVisionHD.close() # Close communications interface
    sys.exit("Exiting script.")

# =====
# Do Something with data... save, export, additional analysis...
# =====
def do_something_with_data():

    # For example, make a peak-peak voltage measurement on channel 1:
    Vpp_Ch1 = \
        str(InfiniiVisionHD.query("MEASure:VPP? CHANnel1")).strip("\n")
    # The result has a newline, so remove it with .strip("\n")
    print("Vpp Ch1 = " + Vpp_Ch1 + " V\n")

# =====
# Main code
# =====

# Connect and initialize oscilloscope
# -----
# Define VISA Resource Manager & Install directory

```

```

rm = pyvisa.ResourceManager('C:\\Windows\\System32\\visa64.dll')

# Define and open the oscilloscope using the VISA address
try:
    InfiniiVisionHD = rm.open_resource(VISA_ADDRESS)
except Exception:
    print("Unable to connect to oscilloscope at " + str(VISA_ADDRESS) \
          + ". Aborting script.\n")
    sys.exit()

# Set the Global Timeout
InfiniiVisionHD.timeout = GLOBAL_TOUT

# Clear the instrument bus
InfiniiVisionHD.clear()

# Clear all status registers and errors
InfiniiVisionHD.write("*CLS")

try:

    # Set up the oscilloscope
    # -----
    # Note that you would normally perform a reset (default setup) if
    # you were to create the setup from scratch... But here we will
    # use the oscilloscope "as is" for the most part.
    # InfiniiVisionHD.query("*RST;*OPC?") # Resets the oscilloscope

    # Always stop the oscilloscope when making any changes.
    InfiniiVisionHD.query(":STOP;*OPC?")

    # Always use normal trigger sweep, never auto.
    InfiniiVisionHD.write(":TRIGger:SWEep NORMal")

    # For this example, the oscilloscope will trigger on CHANnel1.
    # You can use the Probe Comp input signal.
    InfiniiVisionHD.query(":TRIGger:EDGE:SOURce CHANnel1;*OPC?")

    # Clear the display (so you can see the waveform being acquired -
    # otherwise, there is no need for this).
    InfiniiVisionHD.write(":DISPlay:CLEar")

    # Calculate acquisition timeout/wait time by short, overestimate
    # method
    # -----

    # Create some default variables
    N_AVERAGES = 1
    N_SEGMENTS = 1

    # Get some info about the scope time base setup
    HO = float(InfiniiVisionHD.query(":TRIGger:HOLDoff?"))
    T_RANGE = float(InfiniiVisionHD.query(":TIMEbase:RANGe?"))
    T_POSITION = float(InfiniiVisionHD.query(":TIMEbase:POSition?"))

    # Determine Acquisition Type and Mode:
    ACQ_TYPE = str(InfiniiVisionHD.query(":ACQuire:TYPE?").strip("\n"))

```

```

ACQ_MODE = str(InfiniiVisionHD.query(":ACQUIRE:MODE?").strip("\n"))

if ACQ_MODE == "SEGM":
    N_SEGMENTS = \
        float(InfiniiVisionHD.query(":ACQUIRE:SEGMENTED:COUNT?"))
    # Note that if there are a lot of analysis associated segments,
    # for example, serial data decode, the timeout will likely need
    # to be longer than calculated.
    #
    # You are encouraged to manually set up the oscilloscope in
    # this case, as it will be used, time it, and use that, with
    # a little overhead.
    #
    # Blocking method is recommended for Segmented Memory mode.
elif ACQ_TYPE == "AVER":
    N_AVERAGES = float(InfiniiVisionHD.query(":ACQUIRE:COUNT?"))

# Calculate acquisition timeout by overestimate method:
# Recall that PyVISA timeouts are in ms, so multiply by 1000.
SCOPE_ACQUISITION_TIME_OUT = (float(TIME_TO_TRIGGER)*1.1 +
    (T_RANGE*2.0 + abs(T_POSITION)*2.0 + HO*1.1 +
    float(TIME_BETWEEN_TRIGGERS)*1.1)*N_SEGMENTS*N_AVERAGES)*1000.0

## Ensure the timeout is no less than 10 seconds
if SCOPE_ACQUISITION_TIME_OUT < 10000.0:
    SCOPE_ACQUISITION_TIME_OUT = 10000.0

# Acquire Signal
# -----
# Choose blocking_method or polling_method. These were defined as
# functions in case you want to use them repeatedly.
blocking_method()
# If Acquisition Type is Average, always use blocking_method() to
# get complete average.
do_something_with_data()

polling_method()
do_something_with_data()

# Done - cleanup
InfiniiVisionHD.clear() # Clear communications interface
InfiniiVisionHD.close() # Close communications interface
except KeyboardInterrupt:
    InfiniiVisionHD.clear()
    InfiniiVisionHD.query(":STOP;*OPC?")
    InfiniiVisionHD.write(":SYSTEM:LOCK OFF")
    InfiniiVisionHD.clear()
    InfiniiVisionHD.close()
    sys.exit("User Interrupt. Properly closing oscilloscope and "
        "aborting script.")
except Exception:
    InfiniiVisionHD.clear()
    InfiniiVisionHD.query(":STOP;*OPC?")
    InfiniiVisionHD.write(":SYSTEM:LOCK OFF")
    InfiniiVisionHD.clear()
    InfiniiVisionHD.close()
    sys.exit("Something went wrong. Properly closing oscilloscope ")

```

```
        "and aborting script.")  
  
# End of Script  
# -----  
print("Done.")
```


48 More About Oscilloscope Commands

Command Classifications / 1708

Valid Command/Query Strings / 1709

Query Return Values / 1716

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 1708
- **"Non-Core Commands"** on page 1708
- **"Obsolete Commands"** on page 1708

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:

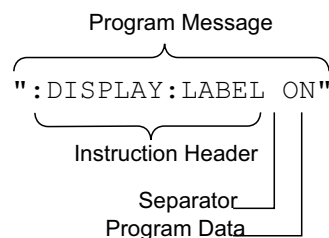
- **Chapter 44**, "Obsolete and Discontinued Commands," starting on page 1647

Valid Command/Query Strings

- **"Program Message Syntax"** on page 1709
- **"Duplicate Mnemonics"** on page 1714
- **"Tree Traversal Rules and Multiple Commands"** on page 1714

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 1710), 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 1711
- **"Compound Command Headers"** on page 1711
- **"Common Command Headers"** on page 1711

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 1712 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 Units Within a program message, there can be multiple *program message units* separated by semicolons (;). Each program message unit represents a separate command or query. See **"Multiple Program Message Units"** on page 1713.

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-Of-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
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.

Multiple Program Message Units

Within a program message, there can be multiple program message units separated by semicolons (;). Each program message unit represents a separate command or query. The program message syntax is:

```
<program message unit>;<program message unit><terminator>
```

For example, in the program message ":TIMEbase:REference CENTER;;DISPlay:LABel ON", one program message unit is ":TIMEbase:REference CENTER" and the other program message unit is ":DISPlay:LABel ON".

Some commands and queries are not supported in program messages that have multiple program message units (for various reasons). Sending these commands and queries within multiple program message units will give unexpected errors.

Commands Not Supported in Multiple Program Message Units

These commands are not supported in program messages that have multiple program message units:

- ":BUS<n>:BITS" on page 316
- ":BUS<n>:MASK" on page 321
- ":MEASure:FFT:THD" on page 692
- ":MEASure:TEDGe" on page 742
- ":RECall:ARBitrary[:START]" on page 944
- ":RECall:DBC[:START]" on page 945
- ":RECall:LDF[:START]" on page 947
- ":RECall:WMEMory<r>[:START]" on page 951
- ":SAVE:ARBitrary[:START]" on page 957
- ":SBUS<n>:IIC:TRIGger:PATtern:DATA" on page 1036
- ":SBUS<n>:IIC:TRIGger:PATtern:DATa2" on page 1037
- ":SBUS<n>:UART:FRAMing" on page 1124
- ":TRIGger:PATtern" on page 1523
- ":WGEN<w>:ARBitrary:DATA" on page 1597

Queries Not Supported in Multiple Program Message Units

These queries are not supported in program messages that have multiple program message units:

- :BUS<n>:BITS? (see [page 316](#))
- :CHANnel<n>:PROBe:ID? (see [page 366](#))
- :MEASure:FFT:THD? (see [page 692](#))

- **":MEASure:RESults?"** on page 724
- **":MEASure:TEDGe?"** (see **page 743**)
- **":SERial?"** on page 276
- **":SYSTem:DIDentifier?"** on page 1426
- **":SYSTem:HID?"** on page 1435

See Also • **"Tree Traversal Rules and Multiple Commands"** on page 1714

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:LABel ON"
```

NOTE

The leading colon before DISPlay:LABel ON tells the parser to go back to the root of the command tree. The parser can then see the DISPlay:LABel ON command. The space between REFeRence and CENTer is required; so is the space between LABel 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.

49 Programming Examples

VISA COM Examples / 1718

VISA Examples / 1752

VISA.NET Examples / 1789

SICL Examples / 1808

SCPI.NET Examples / 1818

The example programs in this manual are ASCII text files that can be cut from the help file and pasted into your favorite text editor.

- See Also**
- You can find additional programming examples for the InfiniiVision HD3-Series oscilloscopes on the Keysight Technologies website at:
www.keysight.com/find/HD3-Series-examples

VISA COM Examples

- ["VISA COM Example in Visual Basic"](#) on page 1718
- ["VISA COM Example in C#" on page 1727](#)
- ["VISA COM Example in Visual Basic .NET"](#) on page 1736
- ["VISA COM Example in Python"](#) on page 1745

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 5.14 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.
#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If
```

```

'
' 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::0x2A8D::0x4704::Unset::0::INSTR")
    myScope.IO.Clear    ' Clear the interface.
    myScope.IO.Timeout = 20000    ' 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.
    ' -----
    Dim byteSetupSave() As Byte
    byteSetupSave = DoQueryIEEEBlock_UI1(":SYSTem:SETup?")

    ' Output file name:
    Dim strOutPath As String
    strOutPath = "c:\scope\config\setup.scp"

    ' If file exists, delete it.
    If (Dir(strOutPath) <> "") Then
        Kill strOutPath
    End If

    ' Output setup string to a file:
    Dim hFileOut As Integer
    hFileOut = FreeFile
    Open strOutPath For Binary Access Write Lock Write As hFileOut
    Put hFileOut, , byteSetupSave ' Write data.
    Close hFileOut ' Close file.

    ' Display number of bytes saved.
    Dim nOutLen As Long
    nOutLen = UBound(byteSetupSave) - LBound(byteSetupSave) + 1
    Debug.Print "Setup bytes saved: " + CStr(nOutLen)

    ' 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, or AVERage).
DoCommand ":ACQuire:TYPE NORMal"
Debug.Print "Acquire type: " + _
    DoQueryString(":ACQuire:TYPE?")

' Or, configure by loading a previously saved setup.
' -----
Dim byteSetupRestore() As Byte

' Output file name:
Dim strInPath As String
strInPath = "c:\scope\config\setup.scp"

' Get setup string from file:
Dim hFileIn As Integer
hFileIn = FreeFile
Open strInPath For Binary Access Read As hFileIn
ReDim byteSetupRestore(0 To LOF(hFileIn) - 1)
Get hFileIn, , byteSetupRestore ' Read data.
Close hFileIn ' Close file.

' Write setup string back to oscilloscope using ":SYSTem:SETup"
' command:
DoCommandIEEEBlock ":SYSTem:SETup", byteSetupRestore

' Display number of bytes restored.
Dim nInLen As Long
nInLen = UBound(byteSetupRestore) - LBound(byteSetupRestore) + 1
Debug.Print "Setup bytes restored: " + CStr(nInLen)

' 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:VAMPlitude"
    varQueryResult = DoQueryNumber(":MEASure:VAMPlitude?")
    MsgBox "Vertical amplitude:" + vbCrLf + _
        FormatNumber(varQueryResult, 4) + " V"

    ' Download the screen image.
    ' -----
    ' Get screen image.
    Dim byteScreen() As Byte
    byteScreen = DoQueryIEEEBlock_UI1(":DISPlay:DATA? PNG")

    ' 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, , byteScreen ' Write data.
    Close hFile ' Close file.
    MsgBox "Screen image (" + CStr(UBound(byteScreen) + 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.
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 WORD"
Debug.Print "Waveform format: " + _
    DoQueryString(":WAVeform:FORMat?")

' Specify the byte order in the word format data:
DoCommand ":WAVeform:BYTeorder LSBF"
Debug.Print "Waveform byte order: " + _
    DoQueryString(":WAVeform:BYTeorder?")

' 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) / 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
Dim lngI As Long

For lngI = 0 To UBound(varQueryResult) Step 2
    lngDataValue = varQueryResult(lngI) + _
        (varQueryResult(lngI + 1) * 256)

    ' Write time value, voltage value.
    Print #hFile, _
        FormatNumber(dblXOrigin + ((lngI / 2) * dblXIncrement), 9) + _
        ", " + _

```



```

        FormatNumber(((lngDataValue - lngYReference) * _
            sngYIncrement) + sngYOrigin)

    Next lngI

    ' Close output file.
    Close hFile ' Close file.
    MsgBox "Waveform format WORD 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.WriteIEEEBlock 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 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual C#, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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 5.14 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 Reference Manager dialog box, select the **COM** hierarchy on the left.
 - d Select **VISA COM 5.14 Type Library**; then click **OK**.
- 6 In the Solution Explorer, expand References, and select **VisaComLib**. Then, in the Properties window, change **Embed Interop Types** to **False**.
- 7 Build and run the program.

For more information, see the VISA COM Help that comes with Keysight IO Libraries Suite.

```

/*
 * 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::0x2A8D::0x4704::Unset::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.scp";
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, or AVERage).
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.scp";
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.
    // -----

    // Get the screen data.
    ResultsArray =
        myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG");
    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 WORD");
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMat?"));

// Specify the byte order in the word format data:
myScope.DoCommand(":WAVEform:BYTeorder LSBF");
Console.WriteLine("Waveform byte order: {0}",
    myScope.DoQueryString(":WAVEform:BYTeorder?"));

// 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");
}

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 / 2);

// 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+=2)
    writer.WriteLine("{0:f9}, {1:f6}",
        fXorigin + ((float)(i / 2) * fXincrement),
        (((float)BitConverter.ToUInt16(ResultsArray, i) - fYreference
e)          * fYincrement) + fYorigin);

// Close output file.
writer.Close();
Console.WriteLine("Waveform format WORD 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 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual Basic, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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 a reference to the VISA COM 5.14 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 Reference Manager dialog box, select the **COM** hierarchy on the left.
 - d Select **VISA COM 5.14 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 In the Solution Explorer, expand References, and select **VisaComLib**. Then, in the Properties window, change **Embed Interop Types** to **False**.
- 7 Build and run the program.

For more information, see the VISA COM Help that comes with Keysight IO Libraries Suite.

```
'
' 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::0x2A8D::0x4704::Unset::0::INSTR")
        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.scp"
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, or AVERage).
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.scp"
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.
    ' -----

    ' Get the screen data.
    ResultsArray = myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG")
    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.
    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 WORD")
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMat?"))

' Specify the byte order in the word format data:
myScope.DoCommand(":WAVEform:BYTeorder LSBF")
Console.WriteLine("Waveform byte order: {0}",
    myScope.DoQueryString(":WAVEform:BYTeorder?"))

' 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")
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 / 2)

' 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 Step 2
    ' Write time value, voltage value.
    writer.WriteLine("{0:f9}, {1:f6}",
        fXorigin + (CSng(index / 2) * fXincrement),
        ((CSng(BitConverter.ToUInt16(ResultsArray, index)) - fYreference) -
        * fYincrement) + fYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format WORD 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()

        ' 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 =
        CDb1(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}': ",
                    strCommand)
                bFirstError = False
            End If
            Console.WriteLine(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 <https://pypi.org/project/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

#!python3
# -*- coding: utf-8 -*-

# 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 sys
import array

```

```

from ctypes.client import GetModule
from ctypes.client import CreateObject

# Run GetModule once to generate ctypes.gen.VisaComLib.
if not hasattr(sys, "frozen"):
    GetModule("C:\Program Files (x86)\IVI Foundation\VISA\VisaCom\
GlobMgr.dll")

import ctypes.gen.VisaComLib as VisaComLib

# Global variables.
# -----
visa_address = "USB0::0x2A8D::0x4704::Unset::0::INSTR"
input_channel = "CHANnel1"
# input_channel = "CHANnel2"
setup_file_name = "setup.scp"
screen_image_file_name = "screen_image.png"
waveform_data_file_name = "waveform_data.csv"

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.
    idn_string = do_query_string("*IDN?")
    print(f"Identification string: '{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(f"Trigger mode: {qresult}")

    # Set EDGE trigger parameters.
    do_command(f":TRIGger:EDGE:SOURce {input_channel}")
    qresult = do_query_string(":TRIGger:EDGE:SOURce?")
    print(f"Trigger edge source: {qresult}")

    do_command(":TRIGger:EDGE:LEVel 1.5")
    qresult = do_query_string(":TRIGger:EDGE:LEVel?")

```

```

print(f"Trigger edge level: {qresult}")

do_command(":TRIGger:EDGE:SLOPe POSitive")
qresult = do_query_string(":TRIGger:EDGE:SLOPe?")
print(f"Trigger edge slope: {qresult}")

# Save oscilloscope setup.
setup_bytes = do_query_ieee_block(":SYSTem:SETup?")
setup_bytes_length = len(setup_bytes)
f = open(setup_file_name, "wb")
f.write(bytearray(setup_bytes))
f.close()
print(f"Setup bytes saved: {setup_bytes_length}")

# Change oscilloscope settings with individual commands:

# Set vertical scale and offset.
do_command(f":{input_channel}:SCALE 0.05")
qresult = do_query_string(f":{input_channel}:SCALE?")
print(f"{input_channel} vertical scale: {qresult}")

do_command(f":{input_channel}:OFFSet -1.5")
qresult = do_query_string(f":{input_channel}:OFFSet?")
print(f"{input_channel} offset: {qresult}")

# Set horizontal scale and offset.
do_command(":TIMEbase:SCALE 0.0002")
qresult = do_query_string(":TIMEbase:SCALE?")
print(f"Timebase scale: {qresult}")

do_command(":TIMEbase:POSition 0.0")
qresult = do_query_string(":TIMEbase:POSition?")
print(f"Timebase position: {qresult}")

# Set the acquisition type.
do_command(":ACQuire:TYPE NORMal")
qresult = do_query_string(":ACQuire:TYPE?")
print(f"Acquire type: {qresult}")

# Or, set up oscilloscope by loading a previously saved setup.
f = open(setup_file_name, "rb")
setup_bytes = f.read()
f.close()
do_command_ieee_block(":SYSTem:SETup", array.array('B', setup_bytes))
print(f"Setup bytes restored: {len(setup_bytes)}")

# Capture an acquisition using :DIGitize.
do_command(f":DIGitize {input_channel}")

# =====
# Analyze:
# =====
def analyze():

    # Make measurements.
    # -----

```

```

do_command(f":MEASure:SOURce {input_channel}")
qresult = do_query_string(":MEASure:SOURce?")
print(f"Measure source: {qresult}")

do_command(":MEASure:FREQuency")
qresult = do_query_string(":MEASure:FREQuency?")
print(f"Measured frequency on {input_channel}: {qresult}")

do_command(":MEASure:VAMPlitude")
qresult = do_query_string(":MEASure:VAMPlitude?")
print(f"Measured vertical amplitude on {input_channel}: {qresult}")

# Download the screen image.
# -----

screen_bytes = do_query_ieee_block(":DISPlay:DATA? PNG")

# Save display data values to file.
f = open(screen_image_file_name, "wb")
f.write(bytearray(screen_bytes))
f.close()
print(f"Screen image written to {screen_image_file_name}.")

# Download waveform data.
# -----

# Set the waveform source.
do_command(f":WAVEform:SOURce {input_channel}")
qresult = do_query_string(":WAVEform:SOURce?")
print(f"Waveform source: {qresult}")

# Set the waveform points mode.
do_command(":WAVEform:POINts:MODE RAW")
qresult = do_query_string(":WAVEform:POINts:MODE?")
print(f"Waveform points mode: {qresult}")

# Get the number of waveform points available.
do_command(":WAVEform:POINts 10240")
qresult = do_query_string(":WAVEform:POINts?")
print(f"Waveform points available: {qresult}")

# Choose the format of the data returned (BYTE or WORD):
do_command(f":WAVEform:FORMat WORD")
qresult = do_query_string(":WAVEform:FORMat?")
print(f"Waveform format: {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",
}

```



```

(
    wav_form,
    acq_type,
    wfmppts,
    avgcnt,
    x_increment,
    x_origin,
    x_reference,
    y_increment,
    y_origin,
    y_reference
) = do_query_numbers(":WAVEform:PREamble?")

print(f"Waveform format: {wav_form_dict[int(wav_form)]}")
print(f"Acquire type: {acq_type_dict[int(acq_type)]}")
print(f"Waveform points desired: {wfmppts}")
print(f"Waveform average count: {avgcnt}")
print(f"Waveform X reference: {x_reference}")    # Always 0.

# Get numeric values for later calculations.
x_increment = do_query_number(":WAVEform:XINcrement?")
print(f"Waveform X increment: {x_increment}")
x_origin = do_query_number(":WAVEform:XORigin?")
print(f"Waveform X origin: {x_origin}")
y_increment = do_query_number(":WAVEform:YINcrement?")
print(f"Waveform Y increment: {y_increment}")
y_origin = do_query_number(":WAVEform:YORigin?")
print(f"Waveform Y origin: {y_origin}")
y_reference = do_query_number(":WAVEform:YREference?")
print(f"Waveform Y reference: {y_reference}")

# Specify the byte order in WORD data.
do_command(":WAVEform:BYTeorder LSBF")
qresult = do_query_string(":WAVEform:BYTeorder?")
print(f"Waveform byte order for WORD data: {qresult}")

# Get the waveform data bytes.
data_bytes = do_query_ieee_block(":WAVEform:DATA?")
data_bytes_length = len(data_bytes)
print(f"Byte count: {data_bytes_length}")

# Get the waveform data values.
data_bytes_array = array.array('B', data_bytes)    # Array of unsigned b
ytes.
values = array.array('H')    # Interpret as array of unsigned shorts.
values.frombytes(data_bytes_array)
print(f"Number of data values: {len(values)}")

# Save waveform data values to CSV file.
f = open(waveform_data_file_name, "w")

for i in range(0, len(values) - 1):
    time_val = x_origin + (i * x_increment)
    voltage = ((values[i] - y_reference) * y_increment) + y_origin
    f.write(f"{time_val:E}, {voltage:f}\n")

```

```

# Close output file.
f.close()
print(f"Waveform format WORD data written to {waveform_data_file_name}.")
")

# =====
# Send a command and check for errors:
# =====
def do_command(command):
    myScope.WriteString(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(query, True)
    result = myScope.ReadString()
    check_instrument_errors(query)
    return result.strip()

# =====
# Send a query, check for errors, return string:
# =====
def do_query_ieee_block(query):
    myScope.WriteString(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(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(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(f"ERROR: {error_string}, command: '{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(visa_address)

# 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()

myScope.IO.Close()
print("End of program.")
sys.exit()

```

VISA Examples

- **"VISA Example in C"** on page 1752
- **"VISA Example in C#"** on page 1761
- **"VISA Example in Visual Basic .NET"** on page 1772
- **"VISA Example in Python"** on page 1782

VISA Example in C

To compile and run this example in Microsoft Visual Studio 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual C++, Windows, Console, Empty project.
- 3 Cut-and-paste the code that follows into a file named "example.c" in the project directory.
- 4 In Visual Studio 2019, right-click the Source Files folder, choose **Add > Add Existing Item...**, select the example.c file, and click **Add**.
- 5 Edit the program to use the VISA address of your oscilloscope.
- 6 In the Solution Explorer, right-click the project (not the solution), and choose **Properties**. In the Property Pages dialog box, 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 Under Configuration Properties, C/C++, Preprocessor, add "_CRT_SECURE_NO_WARNINGS" to the Preprocessor Definitions.
 - d Under Configuration Properties, VC++ Directories, show directories for **Include Directories**, and add the include directory (for example, Program Files (x86)\IVI Foundation\VISA\WinNT\Include).
 - e Under Configuration Properties, VC++ Directories, show directories for **Library Directories**, and add the library files directory (for example, Program Files (x86)\IVI Foundation\VISA\WinNT\lib\msc).
 - f Click **OK** to close the Property Pages dialog.
- 7 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::0x2A8D::0x4704::Unset::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.scp", "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, or AVERage). */
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.scp", "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:VAMPlitude");
do_query_number(":MEASure:VAMPlitude?");
printf("Vertical amplitude: %.2f V\n", num_result);

/* Download the screen image.
 * ----- */

/* Read screen image. */
num_bytes = do_query_ieeeblock(":DISPlay:DATA? PNG");
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 WORD");
do_query_string(":WAVeform:FORMat?");
printf("Waveform format: %s\n", str_result);

/* Specify the byte order in the word format data: */
do_command(":WAVeform:BYTeorder LSBF");
do_query_string(":WAVeform:BYTeorder?");
printf("Waveform byte order: %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");
}

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 / 2);

/* 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+=2)
{
    /* Write time value, voltage value. */
    fprintf(fp, "%9f, %6f\n",
        x_origin + ((float)(i / 2) * x_increment),
        (((float)(ieeeblock_data[i] + (ieeeblock_data[i+1] * 256))
        - y_reference) * y_increment) + y_origin);
}

/* Close output file. */
fclose(fp);
printf("Waveform format WORD 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 C#

To compile and run this example in Microsoft Visual Studio 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual C#, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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 Choose **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.

```

/*
 * 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::0x2A8D::0x4704::Unset::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.scp";
    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, or AVERage).
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.scp";
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.
// -----

// Get the screen data.
nLength = myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG",
    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 WORD");
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMat?"));

// Specify the byte order in the word format data:
myScope.DoCommand(":WAVEform:BYTeorder LSBF");
Console.WriteLine("Waveform byte order: {0}",
    myScope.DoQueryString(":WAVEform:BYTeorder?"));

// 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");
}

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 / 2);

```

```

        // 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+=2)
            writer.WriteLine("{0:f9}, {1:f6}",
                fXorigin + ((float)(i / 2) * fXincrement),
                (((float)BitConverter.ToUInt16(ResultsArray, i) - fYreferenc
e) *
                    fYincrement) + fYorigin);

        // Close output file.
        writer.Close();
        Console.WriteLine("Waveform format WORD 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 VisaGetResultString()
{
    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[3000000];
    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 = 3000000;

    // 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 = VisaGetResultString();

    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 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 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual Basic, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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 > 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.VisaInstrumentApp" 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.

```

'
' 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::0x2A8D::0x4704::Unset::0::INSTR")
      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
      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.scp"
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, or AVERAGE).
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.scp"
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.
    ' -----

    ' Get the screen data.
    nLength = myScope.DoQueryIEEEBlock(":DISPlay:DATA? PNG",
        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 WORD")
Console.WriteLine("Waveform format: {0}",
    myScope.DoQueryString(":WAVEform:FORMat?"))

' Specify the byte order in the word format data:
myScope.DoCommand(":WAVEform:BYTeorder LSBF")
Console.WriteLine("Waveform byte order: {0}",
    myScope.DoQueryString(":WAVEform:BYTeorder?"))

' 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")
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 / 2)

' 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 Step 2
    ' Write time value, voltage value.
    writer.WriteLine("{0:f9}, {1:f6}",
        fXorigin + (CSng(index / 2) * fXincrement),
        ((CSng(BitConverter.ToUInt16(ResultsArray, index)) - fYrefer
ence) -
        * fYincrement) + fYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format WORD 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()

    ' 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 VisaGetResultString() 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(2999999) {}
    Dim length As Integer
    ' Number of bytes returned from instrument.
    ' Set the default number of bytes that will be contained in

```



```

' the ResultsArray to 3,000,000 (~3MB).
length = 3000000

' 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

    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

You can use the Python programming language with the PyVISA package to control Keysight InfiniiVision Series oscilloscopes.

The Python language and PyVISA package can be downloaded from the web at <http://www.python.org/> and <http://pyvisa.readthedocs.io/>, 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

#!/python3
# -*- coding: utf-8 -*-

# *****
# This program illustrates a few commonly-used programming
# features of your Keysight oscilloscope.
# *****

# Import modules.
# -----
import pyvisa
import struct

```

```

import sys

# Global variables.
# -----
visa_address = "USB0::0x2A8D::0x4704::Unset::0::INSTR"
input_channel = "CHANnel1"
# input_channel = "CHANnel2"
setup_file_name = "setup.scp"
screen_image_file_name = "screen_image.png"
waveform_data_file_name = "waveform_data.csv"
debug = False

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.
    idn_string = do_query_string("*IDN?")
    print(f"Identification string: '{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(f"Trigger mode: {qresult}")

    # Set EDGE trigger parameters.
    do_command(f":TRIGger:EDGE:SOURce {input_channel}")
    qresult = do_query_string(":TRIGger:EDGE:SOURce?")
    print(f"Trigger edge source: {qresult}")

    do_command(":TRIGger:EDGE:LEVel 1.5")
    # do_command(":TRIGger:EDGE:LEVel -0.008")
    qresult = do_query_string(":TRIGger:EDGE:LEVel?")
    print(f"Trigger edge level: {qresult}")

    do_command(":TRIGger:EDGE:SLOPe POSitive")
    qresult = do_query_string(":TRIGger:EDGE:SLOPe?")
    print(f"Trigger edge slope: {qresult}")

    # Save oscilloscope setup.
    setup_bytes = do_query_ieee_block(":SYSTem:SETup?")

```

```

f = open(setup_file_name, "wb")
f.write(setup_bytes)
f.close()
print(f"Setup bytes saved: {len(setup_bytes)}")

# Change oscilloscope settings with individual commands:

# Set vertical scale and offset.
do_command(f":{input_channel}:SCALE 0.05")
# do_command(f":{input_channel}:SCALE 0.5")
qresult = do_query_string(f":{input_channel}:SCALE?")
print(f"{input_channel} vertical scale: {qresult}")

do_command(f":{input_channel}:OFFSET -1.5")
# do_command(f":{input_channel}:OFFSET -0.008")
qresult = do_query_string(f":{input_channel}:OFFSET?")
print(f"{input_channel} offset: {qresult}")

# Set horizontal scale and offset.
do_command(":TIMEbase:SCALE 0.0002")
qresult = do_query_string(":TIMEbase:SCALE?")
print(f"Timebase scale: {qresult}")

do_command(":TIMEbase:POSITION 0.0")
qresult = do_query_string(":TIMEbase:POSITION?")
print(f"Timebase position: {qresult}")

# Set the acquisition type.
do_command(":ACQUIRE:TYPE NORMAL")
qresult = do_query_string(":ACQUIRE:TYPE?")
print(f"Acquire type: {qresult}")

# Or, set up oscilloscope by loading a previously saved setup.
setup_bytes = ""
f = open(setup_file_name, "rb")
setup_bytes = f.read()
f.close()
do_command_ieee_block(":SYSTEM:SETup", setup_bytes)
print(f"Setup bytes restored: {len(setup_bytes)}")

# Capture an acquisition using :DIGitize.
do_command(f":DIGitize {input_channel}")

# =====
# Analyze:
# =====
def analyze():

    # Make measurements.
    # -----
    do_command(f":MEASure:SOURce {input_channel}")
    qresult = do_query_string(":MEASure:SOURce?")
    print(f"Measure source: {qresult}")

    do_command(":MEASure:FREQuency")

```

```

qresult = do_query_string(":MEASure:FREQuency?")
print(f"Measured frequency on {input_channel}: {qresult}")

do_command(":MEASure:VAMPlitude")
qresult = do_query_string(":MEASure:VAMPlitude?")
print(f"Measured vertical amplitude on {input_channel}: {qresult}")

# Download the screen image.
# -----
screen_bytes = do_query_ieee_block(":DISPlay:DATA? PNG")

# Save display data values to file.
f = open(screen_image_file_name, "wb")
f.write(screen_bytes)
f.close()
print(f"Screen image written to {screen_image_file_name}.")

# Download waveform data.
# -----

# Set the waveform source.
do_command(f":WAVEform:SOURce {input_channel}")
qresult = do_query_string(":WAVEform:SOURce?")
print(f"Waveform source: {qresult}")

# Set the waveform points mode.
do_command(":WAVEform:POINts:MODE RAW")
qresult = do_query_string(":WAVEform:POINts:MODE?")
print(f"Waveform points mode: {qresult}")

# Get the number of waveform points available.
do_command(":WAVEform:POINts 10240")
qresult = do_query_string(":WAVEform:POINts?")
print(f"Waveform points available: {qresult}")

# Choose the format of the data returned (BYTE or WORD):
do_command(":WAVEform:FORMat WORD")
qresult = do_query_string(":WAVEform:FORMat?")
print(f"Waveform format: {qresult}")

# Specify the byte order in WORD data.
do_command(":WAVEform:BYTeorder LSBF")
qresult = do_query_string(":WAVEform:BYTeorder?")
print(f"Waveform byte order for WORD data: {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",
}

```

```

preamble_string = do_query_string(":WAVEform:PREamble?")
(
    wav_form, acq_type, wfmpmts, avgcnt, x_increment, x_origin,
    x_reference, y_increment, y_origin, y_reference
) = preamble_string.split(",")

print(f"Waveform format: {wav_form_dict[int(wav_form)]}")
print(f"Acquire type: {acq_type_dict[int(acq_type)]}")
print(f"Waveform points desired: {wfmpmts}")
print(f"Waveform average count: {avgcnt}")
print(f"Waveform X increment: {x_increment}")
print(f"Waveform X origin: {x_origin}")
print(f"Waveform X reference: {x_reference}")    # Always 0.
print(f"Waveform Y increment: {y_increment}")
print(f"Waveform Y origin: {y_origin}")
print(f"Waveform Y reference: {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.
data_bytes = do_query_ieee_block(":WAVEform:DATA?")
data_bytes_length = len(data_bytes)
print(f"Byte count: {data_bytes_length}")

block_points = data_bytes_length / 2

# Unpack or split into list of data values.
values = struct.unpack("%dH" % block_points, data_bytes)
print(f"Number of data values: {len(values)}")

# Save waveform data values to CSV file.
f = open(waveform_data_file_name, "w")

for i in range(0, len(values) - 1):
    time_val = x_origin + (i * x_increment)
    voltage = ((values[i] - y_reference) * y_increment) + y_origin
    f.write(f"{time_val:E}, {voltage:f}\n")

# Close output file.
f.close()
print(f"Waveform format WORD data written to {waveform_data_file_name}.")
")

# =====
# Send a command and check for errors:
# =====
def do_command(command, hide_params=False):

    if hide_params:
        (header, data) = command.split(" ", 1)
        if debug:

```

```

        print(f"Cmd = '{header}'")
    else:
        if debug:
            print(f"Cmd = '{command}'")

    InfiniiVision.write(f"{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(f"Cmb = '{command}'")
    InfiniiVision.write_binary_values(f"{command} ", values, datatype='B')
    check_instrument_errors(command)

# =====
# Send a query, check for errors, return string:
# =====
def do_query_string(query):
    if debug:
        print(f"Qys = '{query}'")
    result = InfiniiVision.query(f"{query}")
    check_instrument_errors(query)
    return result.strip()

# =====
# Send a query, check for errors, return floating-point value:
# =====
def do_query_number(query):
    if debug:
        print(f"Qyn = '{query}'")
    results = InfiniiVision.query(f"{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(f"Qyb = '{query}'")
    result = InfiniiVision.query_binary_values(f"{query}", datatype='s', container=bytes)
    check_instrument_errors(query)
    return result

```

```

# =====
# 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(f"ERROR: {error_string}, command: '{command}'")
                print("Exited because of error.")
                sys.exit(1)

            else:    # "No error"
                break

        else:    # :SYSTem:ERRor? should always return string.
            print(f"ERROR: :SYSTem:ERRor? returned nothing, command: '{command}'")
    )
    print("Exited because of error.")
    sys.exit(1)

# =====
# Main program:
# =====

rm = pyvisa.ResourceManager("C:\\Windows\\System32\\visa64.dll")
InfiniiVision = rm.open_resource(visa_address)

InfiniiVision.timeout = 30000
InfiniiVision.clear()

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

InfiniiVision.close()
print("End of program.")
sys.exit()

```


VISA.NET Examples

These programming examples show how to use the VISA.NET drivers that come with Keysight IO Libraries Suite.

- "VISA.NET Example in C#" on page 1789
- "VISA.NET Example in Visual Basic .NET" on page 1795
- "VISA.NET Example in Python" on page 1801

VISA.NET Example in C#

To compile and run this example in Microsoft Visual Studio 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual C#, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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.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 Reference Manager dialog box, under **Assemblies**, select **Extensions**.
 - d In the "Targeting: .NET Framework 4.7.2" list, select the **Ivi.Visa Assembly** check box; then, click **OK**.
- 6 Build and run the program.

For more information, see the VISA.NET Help that comes with Keysight IO Libraries Suite.

```
/*
 * Keysight VISA.NET Example in C#
 * -----
 * This program illustrates a few commonly used programming
 * features of your Keysight InfiniiVision oscilloscope.
 * -----
 */

using System;
using System.IO;
using System.Collections.Generic;
using System.Text;

using Ivi.Visa;
using Ivi.Visa.FormattedIO;
```

```

namespace Example
{
    class Program
    {
        static void Main(string[] args)
        {
            // Change this variable to the address of your instrument
            string VISA_ADDRESS = "USB0::0x2A8D::0x4704::Unset::0::INSTR";

            // Create a connection (session) to the instrument
            IMessageBasedSession session;
            try
            {
                session = GlobalResourceManager.Open(VISA_ADDRESS) as
                    IMessageBasedSession;
            }
            catch (NativeVisaException visaException)
            {
                Console.WriteLine("Couldn't connect.");
                Console.WriteLine("Error is:\r\n{0}\r\n", visaException);
                Console.WriteLine("Press any key to exit...");
                Console.ReadKey();
                return;
            }

            // Create a formatted I/O object which will help us format the
            // data we want to send/receive to/from the instrument
            MessageBasedFormattedIO myScope =
                new MessageBasedFormattedIO(session);

            // For Serial and TCP/IP socket connections enable the read
            // Termination Character, or read's will timeout
            if (session.ResourceName.Contains("ASRL") ||
                session.ResourceName.Contains("SOCKET"))
                session.TerminationCharacterEnabled = true;

            session.TimeoutMilliseconds = 20000;

            // Initialize - start from a known state.
            // =====
            string strResults;
            FileStream fStream;

            // Get and display the device's *IDN? string.
            myScope.WriteLine("*IDN?");
            strResults = myScope.ReadLine();
            Console.WriteLine("*IDN? result is: {0}", strResults);

            // Clear status and load the default setup.
            myScope.WriteLine("*CLS");
            myScope.WriteLine("*RST");

            // Capture data.
            // =====
            // Use auto-scale to automatically configure oscilloscope.
            myScope.WriteLine(":AUToscale");
        }
    }
}

```

```

// Set trigger mode (EDGE, PULSe, PATtern, etc., and input source.
myScope.WriteLine(":TRIGger:MODE EDGE");
myScope.WriteLine(":TRIGger:MODE?");
strResults = myScope.ReadLine();
Console.WriteLine("Trigger mode: {0}", strResults);

// Set EDGE trigger parameters.
myScope.WriteLine(":TRIGger:EDGE:SOURce CHANnel1");
myScope.WriteLine(":TRIGger:EDGE:SOURce?");
strResults = myScope.ReadLine();
Console.WriteLine("Trigger edge source: {0}", strResults);

myScope.WriteLine(":TRIGger:EDGE:LEVel 1.5");
myScope.WriteLine(":TRIGger:EDGE:LEVel?");
strResults = myScope.ReadLine();
Console.WriteLine("Trigger edge level: {0}", strResults);

myScope.WriteLine(":TRIGger:EDGE:SLOPe POSitive");
myScope.WriteLine(":TRIGger:EDGE:SLOPe?");
strResults = myScope.ReadLine();
Console.WriteLine("Trigger edge slope: {0}", strResults);

// Save oscilloscope configuration.
byte[] ResultsArray; // Results array.
int nLength; // Number of bytes returned from instrument.
string strPath;

// Query and read setup string.
myScope.WriteLine(":SYSTem:SETup?");
ResultsArray = myScope.ReadLineBinaryBlockOfByte();
nLength = ResultsArray.Length;

// Write setup string to file.
strPath = "c:\\scope\\config\\setup.scp";
FileStream = File.Open(strPath, FileMode.Create);
FileStream.Write(ResultsArray, 0, nLength);
FileStream.Close();
Console.WriteLine("Setup bytes saved: {0}", nLength);

// Change settings with individual commands:

// Set vertical scale and offset.
myScope.WriteLine(":CHANnel1:SCALe 0.05");
myScope.WriteLine(":CHANnel1:SCALe?");
strResults = myScope.ReadLine();
Console.WriteLine("Channel 1 vertical scale: {0}", strResults);

myScope.WriteLine(":CHANnel1:OFFSet -1.5");
myScope.WriteLine(":CHANnel1:OFFSet?");
strResults = myScope.ReadLine();
Console.WriteLine("Channel 1 vertical offset: {0}", strResults);

// Set horizontal scale and offset.
myScope.WriteLine(":TIMEbase:SCALe 0.0002");
myScope.WriteLine(":TIMEbase:SCALe?");
strResults = myScope.ReadLine();

```

```

Console.WriteLine("Timebase scale: {0}", strResults);

myScope.WriteLine(":TIMEbase:POSition 0.0");
myScope.WriteLine(":TIMEbase:POSition?");
strResults = myScope.ReadLine();
Console.WriteLine("Timebase position: {0}", strResults);

// Set the acquisition type (NORMal, PEAK, or AVERage).
myScope.WriteLine(":ACQuire:TYPE NORMal");
myScope.WriteLine(":ACQuire:TYPE?");
strResults = myScope.ReadLine();
Console.WriteLine("Acquire type: {0}", strResults);

// Or, configure by loading a previously saved setup.
byte[] dataArray;
int nBytesWritten;

// Read setup string from file.
strPath = "c:\\scope\\config\\setup.scp";
dataArray = File.ReadAllBytes(strPath);
nBytesWritten = dataArray.Length;

// Restore setup string.
myScope.Write(":SYSTem:SETup ");
myScope.WriteBinary(dataArray);
myScope.WriteLine("");
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten);

// Capture an acquisition using :DIGitize.
myScope.WriteLine(":DIGitize CHANnel1");

// Analyze the captured waveform.
// =====

// Make a couple of measurements.
// -----
myScope.WriteLine(":MEASure:SOURce CHANnel1");
myScope.WriteLine(":MEASure:SOURce?");
strResults = myScope.ReadLine();
Console.WriteLine("Measure source: {0}", strResults);

double fResult;
myScope.WriteLine(":MEASure:FREQuency");
myScope.WriteLine(":MEASure:FREQuency?");
fResult = myScope.ReadLineDouble();
Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000);

myScope.WriteLine(":MEASure:VAMPlitude");
myScope.WriteLine(":MEASure:VAMPlitude?");
fResult = myScope.ReadLineDouble();
Console.WriteLine("Vertical amplitude: {0:F2} V", fResult);

// Download the screen image.
// -----

// Get the screen data.
myScope.WriteLine(":DISPlay:DATA? PNG");

```

```

ResultsArray = myScope.ReadLineBinaryBlockOfByte();
nLength = ResultsArray.Length;

// Store the screen data to a file.
strPath = "c:\\scope\\data\\screen.png";
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.WriteLine(":WAVEform:POINts:MODE RAW");
myScope.WriteLine(":WAVEform:POINts:MODE?");
strResults = myScope.ReadLine();
Console.WriteLine("Waveform points mode: {0}", strResults);

// Get the number of waveform points available.
myScope.WriteLine(":WAVEform:POINts 10240");
myScope.WriteLine(":WAVEform:POINts?");
strResults = myScope.ReadLine();
Console.WriteLine("Waveform points available: {0}", strResults);

// Set the waveform source.
myScope.WriteLine(":WAVEform:SOURce CHANnel1");
myScope.WriteLine(":WAVEform:SOURce?");
strResults = myScope.ReadLine();
Console.WriteLine("Waveform source: {0}", strResults);

// Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.WriteLine(":WAVEform:FORMat WORD");
myScope.WriteLine(":WAVEform:FORMat?");
strResults = myScope.ReadLine();
Console.WriteLine("Waveform format: {0}", strResults);

// Specify the byte order in the word format data:
myScope.WriteLine(":WAVEform:BYTeorder MSBF");
myScope.WriteLine(":WAVEform:BYTeorder?");
strResults = myScope.ReadLine();
Console.WriteLine("Waveform byte order: {0}", strResults);

// Display the waveform settings:
double[] fResultsArray;
myScope.WriteLine(":WAVEform:PREamble?");
fResultsArray = myScope.ReadLineListOfDouble();

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");
    }

    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.
    ushort[] ul6ResultsArray; // Results array.
    myScope.WriteLine(":WAVEform:DATA?");
    ul6ResultsArray = myScope.ReadLineBinaryBlockOfUInt16();
    nLength = ul6ResultsArray.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)ul6ResultsArray[i] - fYreference)
        * fYincrement) + fYorigin);

// Close output file.
writer.Close();
Console.WriteLine("Waveform format WORD data written to {0}",
    strPath);

// Close the connection to the instrument
// -----
session.Dispose();

Console.WriteLine("Press any key to exit...");
Console.ReadKey();

    }
}

```

VISA.NET Example in Visual Basic .NET

To compile and run this example in Microsoft Visual Studio 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual Basic, Windows, Console Application project using the **.NET Framework 4.7.2**.
- 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 a reference to the VISA.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 Reference Manager dialog box, under **Assemblies**, select **Extensions**.
 - d In the "Targeting: .NET Framework 4.7.2" list, select the **Ivi.Visa Assembly** check box; then, click **OK**.
- 6 Specify the Startup object:
 - 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 **Properties**.
 - c In the Properties dialog box, under **Application**, select the **Startup object:** field and choose **Sub Main**.
 - d Save your change and close the Properties dialog box.

7 Build and run the program.

For more information, see the VISA.NET driver help that comes with Keysight Command Expert.

```
'
' Keysight VISA.NET Example in VB.NET
' -----
' This program illustrates a few commonly used programming
' features of your Keysight InfiniiVision oscilloscope.
' -----

Imports System
Imports System.IO
Imports System.Collections.Generic
Imports System.Text

Imports Ivi.Visa
Imports Ivi.Visa.FormattedIO

Namespace Example
    Class Program

        Public Shared Sub Main(args As String())
            ' Change this variable to the address of your instrument
            Dim VISA_ADDRESS As String = "USB0::0x2A8D::0x4704::Unset::0::INST
R"

            ' Create a connection (session) to the instrument
            Dim session As IMessageBasedSession
            Try
                session = TryCast(GlobalResourceManager.Open(VISA_ADDRESS),
                    IMessageBasedSession)
            Catch visaException As NativeVisaException
                Console.WriteLine("Couldn't connect.")
                Console.WriteLine("Error is:" & vbCrLf & vbCrLf & "{0}" _
                    & vbCrLf & vbCrLf, visaException)
                Console.WriteLine("Press any key to exit...")
                Console.ReadKey()
                Return
            End Try

            ' Create a formatted I/O object which will help us format the
            ' data we want to send/receive to/from the instrument
            Dim myScope As New MessageBasedFormattedIO(session)

            ' For Serial and TCP/IP socket connections enable the read
            ' Termination Character, or read's will timeout
            If session.ResourceName.Contains("ASRL") OrElse
                session.ResourceName.Contains("SOCKET") Then
                session.TerminationCharacterEnabled = True
            End If

            session.TimeoutMilliseconds = 20000

            ' Initialize - start from a known state.
```



```

' =====
Dim strResults As String
Dim fStream As FileStream

' Get and display the device's *IDN? string.
myScope.WriteLine("*IDN?")
strResults = myScope.ReadLine()
Console.WriteLine("*IDN? result is: {0}", strResults)

' Clear status and load the default setup.
myScope.WriteLine("*CLS")
myScope.WriteLine("*RST")

' Capture data.
' =====
' Use auto-scale to automatically configure oscilloscope.
myScope.WriteLine(":AUToscale")

' Set trigger mode (EDGE, PULSe, PATtern, etc., and input source.
myScope.WriteLine(":TRIGger:MODE EDGE")
myScope.WriteLine(":TRIGger:MODE?")
strResults = myScope.ReadLine()
Console.WriteLine("Trigger mode: {0}", strResults)

' Set EDGE trigger parameters.
myScope.WriteLine(":TRIGger:EDGE:SOURce CHANnel1")
myScope.WriteLine(":TRIGger:EDGE:SOURce?")
strResults = myScope.ReadLine()
Console.WriteLine("Trigger edge source: {0}", strResults)

myScope.WriteLine(":TRIGger:EDGE:LEVel 1.5")
myScope.WriteLine(":TRIGger:EDGE:LEVel?")
strResults = myScope.ReadLine()
Console.WriteLine("Trigger edge level: {0}", strResults)

myScope.WriteLine(":TRIGger:EDGE:SLOPe POSitive")
myScope.WriteLine(":TRIGger:EDGE:SLOPe?")
strResults = myScope.ReadLine()
Console.WriteLine("Trigger edge slope: {0}", strResults)

' Save oscilloscope configuration.
Dim ResultsArray As Byte()
' Results array.
Dim nLength As Integer
' Number of bytes returned from instrument.
Dim strPath As String

' Query and read setup string.
myScope.WriteLine(":SYSTem:SETup?")
ResultsArray = myScope.ReadLineBinaryBlockOfByte()
nLength = ResultsArray.Length

' Write setup string to file.
strPath = "c:\scope\config\setup.scp"
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.WriteLine(":CHANnel1:SCALE 0.05")
myScope.WriteLine(":CHANnel1:SCALE?")
strResults = myScope.ReadLine()
Console.WriteLine("Channel 1 vertical scale: {0}", strResults)

myScope.WriteLine(":CHANnel1:OFFSet -1.5")
myScope.WriteLine(":CHANnel1:OFFSet?")
strResults = myScope.ReadLine()
Console.WriteLine("Channel 1 vertical offset: {0}", strResults)

' Set horizontal scale and offset.
myScope.WriteLine(":TIMEbase:SCALE 0.0002")
myScope.WriteLine(":TIMEbase:SCALE?")
strResults = myScope.ReadLine()
Console.WriteLine("Timebase scale: {0}", strResults)

myScope.WriteLine(":TIMEbase:POSition 0.0")
myScope.WriteLine(":TIMEbase:POSition?")
strResults = myScope.ReadLine()
Console.WriteLine("Timebase position: {0}", strResults)

' Set the acquisition type (NORMAL, PEAK, or AVERAGE).
myScope.WriteLine(":ACquire:TYPE NORMAL")
myScope.WriteLine(":ACquire:TYPE?")
strResults = myScope.ReadLine()
Console.WriteLine("Acquire type: {0}", strResults)

' 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.scf"
dataArray = File.ReadAllBytes(strPath)
nBytesWritten = dataArray.Length

' Restore setup string.
myScope.WriteLine(":SYSTem:SETup ")
myScope.WriteBinary(dataArray)
myScope.WriteLine("")
Console.WriteLine("Setup bytes restored: {0}", nBytesWritten)

' Capture an acquisition using :DIGitize.
myScope.WriteLine(":DIGitize CHANnel1")

' Analyze the captured waveform.
' =====

' Make a couple of measurements.
' -----
myScope.WriteLine(":MEASure:SOURce CHANnel1")
myScope.WriteLine(":MEASure:SOURce?")

```

```

strResults = myScope.ReadLine()
Console.WriteLine("Measure source: {0}", strResults)

Dim fResult As Double
myScope.WriteLine(":MEASure:FREQuency")
myScope.WriteLine(":MEASure:FREQuency?")
fResult = myScope.ReadLineDouble()
Console.WriteLine("Frequency: {0:F4} kHz", fResult / 1000)

myScope.WriteLine(":MEASure:VAMPlitude")
myScope.WriteLine(":MEASure:VAMPlitude?")
fResult = myScope.ReadLineDouble()
Console.WriteLine("Vertical amplitude: {0:F2} V", fResult)

' Download the screen image.
' -----

' Get the screen data.
myScope.WriteLine(":DISPlay:DATA? PNG")
ResultsArray = myScope.ReadLineBinaryBlockOfByte()
nLength = ResultsArray.Length

' Store the screen data to a file.
strPath = "c:\scope\data\screen.png"
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.WriteLine(":WAVEform:POINts:MODE RAW")
myScope.WriteLine(":WAVEform:POINts:MODE?")
strResults = myScope.ReadLine()
Console.WriteLine("Waveform points mode: {0}", strResults)

' Get the number of waveform points available.
myScope.WriteLine(":WAVEform:POINts 10240")
myScope.WriteLine(":WAVEform:POINts?")
strResults = myScope.ReadLine()
Console.WriteLine("Waveform points available: {0}", strResults)

' Set the waveform source.
myScope.WriteLine(":WAVEform:SOURce CHANnel")
myScope.WriteLine(":WAVEform:SOURce?")
strResults = myScope.ReadLine()
Console.WriteLine("Waveform source: {0}", strResults)

' Choose the format of the data returned (WORD, BYTE, ASCII):
myScope.WriteLine(":WAVEform:FORMat WORD")
myScope.WriteLine(":WAVEform:FORMat?")
strResults = myScope.ReadLine()
Console.WriteLine("Waveform format: {0}", strResults)

```

```

' Specify the byte order in the word format data:
myScope.WriteLine(":WAVEform:BYTeorder MSBF")
myScope.WriteLine(":WAVEform:BYTeorder?")
strResults = myScope.ReadLine()
Console.WriteLine("Waveform byte order: {0}", strResults)

' Display the waveform settings:
Dim fResultsArray As Double()
myScope.WriteLine(":WAVEform:PREamble?")
fResultsArray = myScope.ReadLineListOfDouble()

Dim fFormat As Double = fResultsArray(0)
If fFormat = 0.0 Then
    Console.WriteLine("Waveform format: BYTE")
ElseIf fFormat = 1.0 Then
    Console.WriteLine("Waveform format: WORD")
ElseIf fFormat = 2.0 Then
    Console.WriteLine("Waveform format: ASCii")
End If

Dim fType As Double = fResultsArray(1)
If fType = 0.0 Then
    Console.WriteLine("Acquire type: NORMAl")
ElseIf fType = 1.0 Then
    Console.WriteLine("Acquire type: PEAK")
ElseIf fType = 2.0 Then
    Console.WriteLine("Acquire type: AVERage")
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)

' Read waveform data.
Dim ul6ResultsArray As UShort()
myScope.WriteLine(":WAVEform:DATA?")
ul6ResultsArray = myScope.ReadLineBinaryBlockOfUInt16()

```

```

nLength = ul6ResultsArray.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}",
        fXorigin + (CSng(i) * fXincrement),
        ((CSng(ul6 ResultsArray(i)) - fYreference) _
        * fYincrement) + fYorigin)
Next

' Close output file.
writer.Close()
Console.WriteLine("Waveform format WORD data written to {0}",
    strPath)

' Close the connection to the instrument
' -----
session.Dispose()

Console.WriteLine("Press any key to exit...")
Console.ReadKey()

End Sub
End Class
End Namespace

```

VISA.NET Example in Python

You can use the Python programming language with the "Python.NET" package (or IronPython) to control Keysight oscilloscopes.

The Python language and "Python.NET" package can be downloaded from the web at <http://www.python.org/> and <http://pythonnet.github.io/>, respectively.

To run this example with Python and "Python.NET":

- 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
```

```

#!python3
# -*- coding: utf-8 -*-

# Keysight VISA.NET Example in Python.NET or IronPython
# *****
# This program illustrates a few commonly-used programming
# features of your Keysight InfiniiVision oscilloscope.
# *****

# Import Python modules.
# -----
import sys
sys.path.append("C:\\Program Files\\IVI Foundation\\VISA\\Microsoft.NET\\
\\Framework64\\v2.0.50727\\VISA.NET Shared Components 7.2.0")

# Import .NET modules.
# -----
import clr
clr.AddReference("Ivi.Visa")
from Ivi.Visa import *
from Ivi.Visa.FormattedIO import *
from System import *
from System.IO import *

# Global variables.
# -----
visa_addr = "USB0::0x2A8D::0x4704::Unset::0::INSTR"
io_timeout_ms = 20000
input_channel = "CHANnel1"
# input_channel = "CHANnel2"
setup_file_name = "setup.scp"
screen_image_file_name = "screen_image.png"
waveform_data_file_name = "waveform_data.csv"

# =====
# Check for instrument errors:
# =====
def check_instrument_errors(when):

    errors_found = False
    while True:
        # Keep reading errors until "No error".
        myScope.WriteLine(":SYSTem:ERRor?")
        error_string = myScope.ReadLine().strip()
        if error_string:    # If there is an error string value.

            if error_string.find("+0,", 0, 3) == -1:    # Not "No error".
                errors_found = True
                print(f"ERROR: {error_string}")

            else:    # "No error"
                break

    else:    # :SYSTem:ERRor? should always return string.
        errors_found = True
        print("ERROR: ':SYSTem:ERRor?' empty.")

```

```

        break

if errors_found:
    print(f"Exited because error(s) found when: '{when}'")
    sys.exit(1)

# =====
# Initialize:
# =====
def initialize():

    # Get and display the device's *IDN? string.
    myScope.WriteLine("*IDN?")
    idn_string = myScope.ReadLine().strip()
    print(f"Identification string: '{idn_string}'")

    # Clear status and load the default setup.
    myScope.WriteLine("*CLS")
    myScope.WriteLine("*RST")

# =====
# Capture:
# =====
def capture():

    # Display the input channel.
    myScope.WriteLine(":CHANnel:DISPlay OFF")
    myScope.WriteLine(f":{input_channel}:DISPlay ON")

    # Use auto-scale to automatically set up oscilloscope.
    print("Autoscale.")
    myScope.WriteLine(":AUToscale:CHANnels DISPlayed")
    myScope.WriteLine(":AUToscale")

    # Set trigger mode.
    myScope.WriteLine(":TRIGger:MODE EDGE")
    myScope.WriteLine(":TRIGger:MODE?")
    qresult = myScope.ReadLine().strip()
    print(f"Trigger mode: {qresult}")

    # Set EDGE trigger parameters.
    myScope.WriteLine(f":TRIGger:EDGE:SOURce {input_channel}")
    myScope.WriteLine(":TRIGger:EDGE:SOURce?")
    qresult = myScope.ReadLine().strip()
    print(f"Trigger edge source: {qresult}")

    myScope.WriteLine(":TRIGger:EDGE:LEVel 1.5")
    myScope.WriteLine(":TRIGger:EDGE:LEVel?")
    qresult = myScope.ReadLine().strip()
    print(f"Trigger edge level: {qresult}")

    myScope.WriteLine(":TRIGger:EDGE:SLOPe POSitive")
    myScope.WriteLine(":TRIGger:EDGE:SLOPe?")
    qresult = myScope.ReadLine().strip()
    print(f"Trigger edge slope: {qresult}")

```

```

# Save oscilloscope setup to file.
myScope.WriteLine(":SYSTem:SETup?")
setup_bytes = myScope.ReadLineBinaryBlockOfByte()
File.WriteAllBytes(setup_file_name, setup_bytes)
print(f"Setup bytes saved: {len(setup_bytes)}")

# Change settings with individual commands:

# Set vertical scale and offset.
myScope.WriteLine(f":{input_channel}:SCALe 0.05")
myScope.WriteLine(f":{input_channel}:SCALe?")
qresult = myScope.ReadLine().strip()
print(f"{input_channel} vertical scale: {qresult}")

myScope.WriteLine(f":{input_channel}:OFFSet -1.5")
myScope.WriteLine(f":{input_channel}:OFFSet?")
qresult = myScope.ReadLine().strip()
print(f"{input_channel} offset: {qresult}")

# Set horizontal scale and offset.
myScope.WriteLine(":TIMEbase:SCALe 0.0002")
myScope.WriteLine(":TIMEbase:SCALe?")
qresult = myScope.ReadLine().strip()
print(f"Timebase scale: {qresult}")

myScope.WriteLine(":TIMEbase:POSition 0.0")
myScope.WriteLine(":TIMEbase:POSition?")
qresult = myScope.ReadLine().strip()
print(f"Timebase position: {qresult}")

# Set the acquisition type.
myScope.WriteLine(":ACQuire:TYPE NORMAl")
myScope.WriteLine(":ACQuire:TYPE?")
qresult = myScope.ReadLine().strip()
print(f"Acquire type: {qresult}")

# Or, configure by loading a previously saved setup.

# Read setup string from file.
setup_bytes = File.ReadAllBytes(setup_file_name)

# Restore setup string.
myScope.Write(":SYSTem:SETup ")
write_binary = myScope.WriteBinary.Overloads[Array[Byte]]
write_binary(setup_bytes)
myScope.WriteLine("")
print(f"Setup bytes restored: {len(setup_bytes)}")

# Capture an acquisition using :DIGitize.
myScope.WriteLine(f":DIGitize {input_channel}")

# =====
# Analyze:
# =====
def analyze():

```



```

# Make measurements.
# -----
myScope.WriteLine(f":MEASure:SOURce {input_channel}")
myScope.WriteLine(":MEASure:SOURce?")
qresult = myScope.ReadLine().strip()
print(f"Measure source: {qresult}")

myScope.WriteLine(":MEASure:FREQuency")
myScope.WriteLine(":MEASure:FREQuency?")
qresult = myScope.ReadLineDouble()
print(f"Measured frequency on {input_channel}: {qresult:f}")

myScope.WriteLine(":MEASure:VAMPlitude")
myScope.WriteLine(":MEASure:VAMPlitude?")
qresult = myScope.ReadLineDouble()
print(f"Measured vertical amplitude on {input_channel}: {qresult:f}")

# Download the screen image.
# -----

# Get the screen data.
myScope.WriteLine(":DISPlay:DATA? PNG")
image_bytes = myScope.ReadLineBinaryBlockOfByte()
image_bytes_length = len(image_bytes)
fStream = File.Open(screen_image_file_name, FileMode.Create)
fStream.Write(image_bytes, 0, image_bytes_length)
fStream.Close()
print(f"Screen image written to {screen_image_file_name}.")

# Download waveform data.
# -----

# Set the waveform source.
myScope.WriteLine(f":WAVeform:SOURce {input_channel}")
myScope.WriteLine(":WAVeform:SOURce?")
qresult = myScope.ReadLine().strip()
print(f"Waveform source: {qresult}")

# Set the waveform points mode.
myScope.WriteLine(":WAVeform:POINts:MODE RAW")
myScope.WriteLine(":WAVeform:POINts:MODE?")
qresult = myScope.ReadLine().strip()
print(f"Waveform points mode: {qresult}")

# Get the number of waveform points available.
myScope.WriteLine(":WAVeform:POINts 10240")
myScope.WriteLine(":WAVeform:POINts?")
qresult = myScope.ReadLine().strip()
print(f"Waveform points available: {qresult}")

# Choose the format of the data returned (BYTE or WORD):
myScope.WriteLine(":WAVeform:FORMat WORD")
myScope.WriteLine(":WAVeform:FORMat?")
qresult = myScope.ReadLine().strip()
print(f"Waveform format: {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",
}

myScope.WriteLine(":WAVEform:PREamble?")
(
    wav_form, acq_type, wfmppts, avgcnt, x_increment, x_origin,
    x_reference, y_increment, y_origin, y_reference
) = myScope.ReadLine().split(",")

print(f"Waveform format: {wav_form_dict[int(wav_form)]}")
print(f"Acquire type: {acq_type_dict[int(acq_type)]}")
print(f"Waveform points desired: {wfmppts}")
print(f"Waveform average count: {avgcnt}")
print(f"Waveform X reference: {x_reference}")    # Always 0.

# Get numeric values for later calculations.
myScope.WriteLine(":WAVEform:XINCrement?")
x_increment = myScope.ReadLineDouble()
print(f"Waveform X increment: {x_increment}")
myScope.WriteLine(":WAVEform:XORigin?")
x_origin = myScope.ReadLineDouble()
print(f"Waveform X origin: {x_origin}")
myScope.WriteLine(":WAVEform:YINCrement?")
y_increment = myScope.ReadLineDouble()
print(f"Waveform Y increment: {y_increment}")
myScope.WriteLine(":WAVEform:YORigin?")
y_origin = myScope.ReadLineDouble()
print(f"Waveform Y origin: {y_origin}")
myScope.WriteLine(":WAVEform:YREFerence?")
y_reference = myScope.ReadLineDouble()
print(f"Waveform Y reference: {y_reference}")

# Specify the byte order in WORD data.
myScope.WriteLine(":WAVEform:BYTeorder MSBF")
myScope.WriteLine(":WAVEform:BYTeorder?")
qresult = myScope.ReadLine().strip()
print(f"Waveform byte order for WORD data: {qresult}")

# Get the waveform data values.
myScope.WriteLine(":WAVEform:DATA?")
values = myScope.ReadLineBinaryBlockOfUInt16()
values_length = len(values)
print(f"Number of data values: {values_length}")

# Open file for output.
writer = File.CreateText(waveform_data_file_name)

# Output waveform data in CSV format.

```

```

for i in range(0, values_length - 1):
    time_val = x_origin + i * x_increment
    voltage = (values[i] - y_reference) * y_increment + y_origin
    writer.WriteLine(f"{time_val:E}, {voltage:f}")

# Close output file.
writer.Close()
print(f"Waveform format WORD data written to {waveform_data_file_name}.")
")

# =====
# Main program:
# =====
session = GlobalResourceManager.Open(visa_addr)
session.TimeoutMilliseconds = io_timeout_ms
myScope = MessageBasedFormattedIO(session)

# Initialize the oscilloscope, capture data, and analyze.
initialize()
capture()
analyze()

# Check for oscilloscope instrument errors.
check_instrument_errors("End of program")

# Close the connection to the instrument
session.Dispose()
print("End of program.")

# Wait for a key press before exiting.
print("Press any key to exit...")
Console.ReadKey(True)

```

SICL Examples

- **"SICL Example in C"** on page 1808

SICL Example in C

To compile and run this example in Microsoft Visual Studio 2019:

- 1 Open Visual Studio.
- 2 Create a new Visual C++, Windows, Console, Empty project.
- 3 Cut-and-paste the code that follows into a file named "example.c" in the project directory.
- 4 In Visual Studio 2019, right-click the Source Files folder, choose **Add > Add Existing Item...**, select the example.c file, and click **Add**.
- 5 Change the build target from "x86" to "x64".
- 6 Edit the program to use the VISA address of your oscilloscope.
- 7 In the Solution Explorer, right-click the project (not the solution), and choose **Properties**. In the Property Pages dialog box, 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 Under Configuration Properties, C/C++, Preprocessor, add "_CRT_SECURE_NO_WARNINGS" to the Preprocessor Definitions.
 - d Under Configuration Properties, VC++ Directories, show directories for **Include Directories**, and add the include directory (for example, Program Files\Keysight\IO Libraries Suite\include).
 - e Under Configuration Properties, VC++ Directories, show directories for **Library Directories**, and add the library files directory (for example, Program Files\Keysight\IO Libraries Suite\lib_x64).
 - f Click **OK** to close the Property Pages dialog.
- 8 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[10893::18180::Unset::0]"
#define TIMEOUT           5000
#define IEEEBLOCK_SPACE   3000000

/* 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[IEEBLOCK_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. 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.scp", "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.scp.\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:POSition 0.0");
do_query_string(":TIMebase:POSition?");
printf("Timebase position: %s\n", str_result);

/* Set the acquisition type (NORMAL, PEAK, or AVERage). */
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.scp", "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.scp.\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:VAMPlitude");
        do_query_number(":MEASure:VAMPlitude?");
        printf("Vertical amplitude: %.2f V\n", num_result);

        /* Download the screen image.
        * ----- */

        /* Read screen image. */
        num_bytes = do_query_ieeeblock(":DISPlay:DATA? PNG");
        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 WORD");
do_query_string(":WAVEform:FORMat?");
printf("Waveform format: %s\n", str_result);

/* Specify the byte order in the word format data: */
do_command(":WAVEform:BYTeorder LSBF");
do_query_string(":WAVEform:BYTeorder?");
printf("Waveform byte order: %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");
}

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 / 2);

    /* 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+=2)
    {
        /* Write time value, voltage value. */
        fprintf(fp, "%9f, %6f\n",
            x_origin + ((float)(i / 2) * x_increment),
            (((float)(ieeeblock_data[i] + (ieeeblock_data[i + 1] * 256))
            - y_reference) * y_increment) + y_origin);
    }

    /* Close output file. */
    fclose(fp);
    printf("Waveform format WORD 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");
    iprintf(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");
    iprintf(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");
    iprintf(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");
    iprintf(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);
    }
}

```

```
}
}
```

SCPI.NET Examples

You can also program the oscilloscope using the SCPI.NET drivers that come with Keysight's free Command Expert software.

While you can write code manually using the SCPI.NET drivers, 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.

To download the Keysight Command Expert software, see:

<http://www.keysight.com/find/commandexpert>

For more on programming with the SCPI.NET drivers, see "Using SCPI.NET Drivers" in the help that comes with Keysight Command Expert.

Index

Symbols

*ESR? (Event Status Register) query, [91](#)
*OPC (Operation Complete) query, [91](#)
+9.9E+37, infinity representation, [1716](#)
+9.9E+37, measurement error, [672](#)
+Width to +Width jitter measurement, [681](#)

Numerics

0 (zero) values in waveform data, [1563](#)
1 (one) values in waveform data, [1563](#)
3000G X-Series oscilloscopes, command differences from, [55](#)

A

A429 SEARCh commands, [1204](#)
A429 serial bus commands, [988](#)
absolute value math function, [562](#)
AC coupling, trigger edge, [1506](#)
AC input coupling for specified channel, [350](#)
AC RMS measured on waveform, [763](#)
accumulate activity, [262](#)
ACQuire commands, [283](#)
acquire data, [271](#), [311](#)
acquire mode on autoscale, [267](#)
acquire reset conditions, [247](#), [1445](#)
acquire sample rate, [307](#)
acquire sample rate, setting, [307](#)
ACQuire subsystem, [83](#)
acquired data points, [297](#)
acquired data points, setting, [297](#)
acquisition anti-alias control, [286](#)
acquisition count, [288](#)
acquisition mode, [284](#), [296](#), [1582](#)
acquisition points, max (Digitizer mode), [301](#)
acquisition type, [284](#), [311](#)
acquisition types, [1556](#)
active edges, [262](#)
activity logic levels, [262](#)
activity on digital channels, [262](#)
add function, [1577](#)
add math function, [561](#)
address field size, IIC serial decode, [1032](#)
address value, USB search, [1257](#)
address value, USB trigger, [1145](#)
address, IIC trigger pattern, [1035](#)

Adjacent Channel Power Ratio (ACPR), FFT analysis measurement, [689](#)
AER (Arm Event Register), [263](#)
all (snapshot) measurement, [723](#)
all analog channels, perform color grade analysis on, [335](#)
all edges, measure, [606](#)
ALL segments waveform save option, [979](#)
all segments, :WAVEform:DATA?, [1573](#)
AM depth, waveform generator modulation, [1616](#)
AM modulation type, waveform generator, [1623](#)
amplitude profile, Control Loop Response (Bode) power analysis, [862](#)
amplitude profile, frequency response analysis, [515](#)
amplitude profile, Power Supply Rejection Ratio (PSRR) power analysis, [906](#)
amplitude, vertical, [756](#)
amplitude, waveform generator, [513](#), [1632](#)
analog channel coupling, [350](#)
analog channel display, [351](#)
analog channel impedance, [352](#)
analog channel inversion, [353](#)
analog channel labels, [354](#), [439](#)
analog channel memory depth, automatic, [298](#)
analog channel offset, [355](#)
analog channel protection lock, [1449](#)
analog channel range, [374](#)
analog channel sample rate, automatic, [292](#), [308](#)
analog channel scale, [375](#)
analog channel source for glitch, [1519](#)
analog channel units, [364](#), [376](#)
analog probe attenuation, [356](#)
analog probe head type, [365](#)
analog probe skew, [369](#)
analysis record, [1568](#)
analysis record length, jitter analysis and, [606](#)
analysis results, save, [967](#)
analyze peaks in analysis record, [1444](#)
analyzing captured data, [79](#)
angle brackets, [228](#)
annotate channels, [354](#)
annotation background, display, [411](#)
annotation color, display, [412](#)
annotation grid association, [413](#)
annotation mode, [414](#)
annotation text, display, [416](#)

annotation waveform source association, [415](#)
annotation X1 position, [417](#)
annotation Y position, [418](#)
annotation, display, [410](#)
anti-alias control, [286](#)
apparent power, [775](#)
arbitrary waveform generator output, [1612](#)
arbitrary waveform, byte order, [1596](#)
arbitrary waveform, capturing from other sources, [1604](#)
arbitrary waveform, clear, [1601](#)
arbitrary waveform, download DAC values, [1602](#)
arbitrary waveform, download floating-point values, [1597](#)
arbitrary waveform, interpolation, [1603](#)
arbitrary waveform, points, [1600](#)
arbitrary waveform, recall, [944](#)
arbitrary waveform, save, [957](#)
area measurement, [673](#)
ARINC 429 auto setup, [990](#)
ARINC 429 base, [991](#)
ARINC 429 signal speed, [996](#)
ARINC 429 signal type, [994](#)
ARINC 429 source, [995](#)
ARINC 429 trigger data pattern, [998](#), [1207](#)
ARINC 429 trigger label, [997](#), [1001](#), [1205](#)
ARINC 429 trigger SDI pattern, [999](#), [1208](#)
ARINC 429 trigger SSM pattern, [1000](#), [1209](#)
ARINC 429 trigger type, [1002](#), [1206](#)
ARINC 429 triggering, [983](#)
ARINC 429 user-defined baud rate, [992](#)
ARINC 429 word format, [993](#)
Arm Event Register (AER), [263](#)
Arm Event Register, reading a Condition register bit, [1306](#)
Arm Event Register, reading and clearing an Event register bit, [1310](#)
Arm Event Register, reading and clearing the Event register, [1315](#)
Arm Event Register, reading the Condition register, [1311](#)
Arm Event Register, setting and reading a Negative Transition filter bit, [1308](#)
Arm Event Register, setting and reading a Positive Transition filter bit, [1309](#)
Arm Event Register, setting and reading an Enable register bit, [1307](#)
Arm Event Register, setting and reading the Enable register, [1312](#)

Arm Event Register, setting and reading the Negative Transition filter, 1313
 Arm Event Register, setting and reading the Positive Transition filter, 1314
 ARM status commands, 1303
 armed status, checking for, 1677
 arming edge slope, Edge Then Edge trigger, 1499
 arming edge source, Edge Then Edge trigger, 1500
 ASCII format, 1565
 ASCII format for data transfer, 1558
 ASCII string, quoted, 228
 ASCiixy waveform data format, 976
 assign channel names, 354
 attenuation factor (external trigger) probe, 461
 attenuation for oscilloscope probe, 356
 Auto Range capability for DVM, 454
 auto set up, trigger level, 1491
 auto setup (ARINC 429), 990
 auto setup for M1553 trigger, 1060
 auto setup for power analysis signals, 909
 auto trigger sweep mode, 1479
 automask create, 800
 automask source, 801
 automask units, 802
 automatic measurements constants, 356
 autoscale, 265
 autoscale acquire mode, 267
 autoscale channels, 268
 AUToscale command, 82
 AUX OUT BNC, 327
 average math function, clear, 532
 average value measurement, 757
 Average, power modulation analysis, 885
 averaged value math function, 562
 averaging acquisition type, 284, 1557
 averaging, synchronizing with, 1693
 Ax + B math function, 562

B

backlight, display, 419
 band pass filter math function, 562
 band-pass filter math function, center frequency, 549
 band-pass filter math function, frequency width, 550
 bandwidth, 3
 bandwidth filter limit to 20 MHz, 1656
 bandwidth filter limits, 460
 bar chart of current harmonics results, 869
 BARTlett window, 490, 548, 604
 base 10 exponential math function, 562
 base value measurement, 758
 base, ARINC 429, 991
 base, MIL-STD-1553 serial decode, 1061
 base, SENT decode, 1071
 base, UART trigger, 1129
 base, USB decode, 1139

basic instrument functions, 235
 baud rate, 1010, 1046, 1122
 baud rate, CAN FD, 1012
 baud rate, CAN XL, 1013
 baud rate, user-defined, ARINC 429, 992
 begin acquisition, 271, 275, 277
 BHARis window for minimal spectral leakage, 490, 548, 604
 Bin Size, FFT, 534
 binary block data, 228, 425, 574, 1457, 1563
 BINary waveform data format, 976
 bind levels for masks, 821
 bit order, 1123
 bit order, SPI decode, 1104
 bit rate measurement, 674
 bit selection command, bus, 315
 bit weights, 239
 bitmap display, 425, 575
 bits in Service Request Enable Register, 252
 bits in Standard Event Status Enable Register, 238
 bits in Status Byte Register, 254
 bits selection command, bus, 316
 blank, 270
 block data, 228, 242, 1457
 block response data, 86
 blocking commands, 91
 blocking synchronization, 1687
 blocking synchronization example, 1695
 blocking wait, 1686
 BMP format screen image data, 425, 575
 braces, 227
 built-in measurements, 79
 burst count, WaveGen output, 1626
 burst waveform generator output, 1625
 burst width measurement, 675
 bus bit selection command, 315
 bus bits selection commands, 316
 bus clear command, 318
 bus commands, 314
 BUS data format, 1560
 bus display, 319
 bus label command, 320
 bus mask command, 321
 BUS<n> commands, 313
 button disable, 1436
 byte format for data transfer, 1559, 1565
 BYTEorder, 1561

C

C, SICL library example, 1808
 C, VISA library example, 1752
 C#, VISA COM example, 1727
 C#, VISA example, 1761
 C#, VISA.NET example, 1789
 Cal Protect control, 328
 CAL PROTECT switch, 324
 calculating preshoot of waveform, 721

calculating the waveform overshoot, 715
 calibrate, 325, 326, 332
 CALibrate commands, 323
 calibrate date, 325
 calibrate introduction, 324
 calibrate label, 326
 calibrate output, 327
 calibrate start, 329
 calibrate status, 330
 calibrate temperature, 331
 calibrate time, 332
 calibration, protecting against, 328
 CAN acknowledge, 1009
 CAN baud rate, 1010
 CAN FD baud rate, 1012
 CAN FD data triggers, starting byte position, 1022
 CAN protocol decode type, 1029
 CAN SEARCh commands, 1210
 CAN serial bus commands, 1004
 CAN serial search, data, 1213
 CAN serial search, data length, 1214
 CAN serial search, ID, 1215
 CAN serial search, ID mode, 1216
 CAN serial search, mode, 1211
 CAN signal definition, 1011
 CAN source, 1014
 CAN symbolic data display, 1007
 CAN symbolic data, recall, 945
 CAN trigger, 1015, 1021
 CAN trigger data pattern, 1019
 CAN trigger ID pattern, 1023
 CAN trigger pattern id mode, 1024
 CAN trigger type, 1028
 CAN trigger, ID filter for, 1018
 CAN triggering, 984
 CAN XL baud rate, 1013
 capture data, 271
 capturing data, 78
 cardiac waveform generator output, 1611
 center frequency set, 470, 535, 561
 center of screen, 1590
 center screen voltage, 362
 center screen, FFT vertical value at, 480, 484
 center screen, vertical value at, 560, 566
 center time base reference, 1473
 channel, 281, 354
 channel coupling, 350
 channel display, 351
 channel input impedance, 352
 channel inversion, 353
 channel label, 354
 channel labels, 438, 439
 channel overload, 373
 Channel Power, FFT analysis measurement, 690
 channel protection, 373
 channel reset conditions, 247, 1445
 channel selected to produce trigger, 1519
 channel signal type, 371
 channel skew for oscilloscope probe, 369

- channel status, 278
- channel vernier, 377
- channel, stop displaying, 270
- CHANnel<n> commands, 345, 348
- channels to autoscale, 268
- channels, how autoscale affects, 265
- characters to display, 1427
- chart logic bus state, clock edge, 528
- chart logic bus state, clock source, 527
- chart logic bus, units, 531
- chart logic bus, value for data = 0, 530
- chart logic bus, value for data increment, 529
- classes of input signals, 490, 548
- classifications, command, 1708
- clear, 420
- clear bus command, 318
- clear FFT function, 471
- clear markers, 676, 1657
- clear measurement, 676, 1657
- clear message from display, 440
- clear message queue, 236
- Clear method, 81
- clear persistence data from display, 442
- clear reference waveforms, 1639
- clear status, 236
- clear waveform area, 409
- clipped high waveform data value, 1563
- clipped low waveform data value, 1563
- clock, 1033, 1105, 1109
- clock (oscilloscope) display, 424
- clock (oscilloscope) display location, 421
- clock (oscilloscope) horizontal X position, 422
- clock (oscilloscope) vertical Y position, 423
- clock period, SENT signal, 1069
- clock recovery commands, 379
- clock recovery method, 382
- clock source, setup and hold trigger, 1537
- Clock TIE jitter measurement, 750
- clock timeout, SPI, 1106
- clock, recovered, displaying, 381
- CLS (Clear Status), 236
- CME (Command Error) status bit, 238, 240
- CMOS threshold voltage for digital channels, 404
- code, :ACQuire:COMPLete, 287
- code, :ACQuire:SEGMENTed, 304
- code, :ACQuire:TYPE, 312
- code, :AUToscale, 266
- code, :CHANnel<n>:LABel, 354
- code, :CHANnel<n>:PROBe, 356
- code, :CHANnel<n>:RANGe, 374
- code, :DIGitize, 272
- code, :DISPlay:DATA, 425
- code, :DISPlay:LABel, 438
- code, :MEASure:PERiod, 734
- code, :MEASure:RESults, 724
- code, :MEASure:TEDGe, 743
- code, :MTESt, 795
- code, :POD<n>:NIBBlE<n>:THReshold, 833
- code, :POD<n>:THReshold, 835
- code, :RUN/:STOP, 275
- code, :SYSTem:SETup, 1457
- code, :TIMebase:DELAy, 1659
- code, :TIMebase:MODE, 1472
- code, :TIMebase:RANGe, 1470
- code, :TIMebase:REfERENCE, 1473
- code, :TRIGger:MODE, 1494
- code, :TRIGger:SLOPe, 1509
- code, :TRIGger:SOURce, 1510
- code, :VIEW and :BLANK, 281
- code, :WAVEform, 1578
- code, :WAVEform:DATA, 1563
- code, :WAVEform:POINts, 1567
- code, :WAVEform:PREamble, 1571
- code, :WAVEform:SEGMENTed, 304
- code, :WGEN:ARbitrary:DATA, 1597
- code, *RST, 249
- code, SIDL library example in C, 1808
- code, VISA COM library example in C#, 1727
- code, VISA COM library example in Python, 1745
- code, VISA COM library example in Visual Basic, 1718
- code, VISA COM library example in Visual Basic .NET, 1736
- code, VISA library example in C, 1752
- code, VISA library example in C#, 1761
- code, VISA library example in Python, 1782
- code, VISA library example in Visual Basic .NET, 1772
- code, VISA.NET library example in C#, 1789
- code, VISA.NET library example in Python, 1801
- code, VISA.NET library example in Visual Basic .NET, 1795
- colon, root commands prefixed by, 261
- color grade commands, 333
- color grade key, source when there are multiple channels, 337
- color grade levels display, 339
- color grade reset, 341
- color grade theme, 342
- color grade waveform display, 344
- color grade waveform invert, 336
- color grade waveform source, 343
- Comma Separated Values (CSV) waveform data format, 976
- command classifications, 1708
- command differences from 3000G X-Series oscilloscopes, 55
- command errors detected in Standard Event Status Register, 240
- Command Expert, 1789, 1818
- command header, 1709
- command headers, common, 1711
- command headers, compound, 1711
- command headers, simple, 1711
- command strings, valid, 1709
- commands quick reference, 93
- commands sent over interface, 235
- commands, more about, 1707
- commands, obsolete and discontinued, 1647
- common (*) commands, 3, 231, 235
- common command headers, 1711
- common logarithm math function, 562
- completion criteria for an acquisition, 287, 288
- compound command headers, 1711
- compound header, 1714
- computer control examples, 1717
- concurrent commands, 91
- conditions for external trigger, 459
- conditions, reset, 247, 1445
- conduction calculation method for switching loss, 931
- configurations, oscilloscope, 242, 246, 250, 1457
- connect oscilloscope, 71
- Connection Expert, 73
- constant frequency clock recovery, 382
- constants for making automatic measurements, 356
- constants for scaling display factors, 356
- constants for setting trigger levels, 356
- Control Loop Response (Bode) power analysis settings, 845
- control loop response (Bode) power analysis, apply, 846
- control loop response power analysis, single frequency, 853
- control loop response power analysis, sweep start frequency, 854
- control loop response power analysis, sweep stop frequency, 855
- controller initialization, 78
- conversion type of power supply, 882
- copyright, 2
- core commands, 1708
- count, 1562
- count values, 288
- count, averaged value math function, 526
- count, averages, FFT function, 468
- counter, 385, 677
- counter commands, 385
- counter mode, 389
- counter source, 391
- counter totalize, clear, 392
- counter totalize, slope, 393
- counter, digits of resolution, 390
- counter, enable/disable, 388
- coupling, 1506
- coupling for channels, 350
- CRC format, SENT, 1070
- CRC value, USB search, 1258
- CRC value, USB trigger, 1146
- create automask, 800
- crest factor, 777
- CSV (Comma Separated Values) waveform data format, 976
- current harmonics analysis fail count, 870

- current harmonics analysis results, save, [965](#)
 - current harmonics analysis run count, [875](#)
 - current harmonics analysis, apply, [867](#)
 - current harmonics results data, [868](#)
 - current harmonics results display, [869](#)
 - current logic levels on digital channels, [262](#)
 - current oscilloscope configuration, [242](#), [246](#), [250](#), [1457](#)
 - current probe, [364](#), [376](#), [463](#)
 - CURRent segment waveform save option, [979](#)
 - current source, [926](#)
 - cursor display setting, X1, [636](#)
 - cursor display setting, X2, [639](#)
 - cursor display setting, Y1, [645](#)
 - cursor display setting, Y2, [647](#)
 - cursor mode, [635](#)
 - cursor position, [634](#), [637](#), [640](#), [642](#), [646](#), [649](#)
 - cursor reset conditions, [248](#), [1445](#)
 - cursor source, [638](#), [641](#)
 - cursor units, X, [643](#), [644](#)
 - cursor units, Y, [650](#), [651](#)
 - cursor values, saving, [968](#)
 - cursors track measurements, [730](#)
 - cursors, how autoscale affects, [265](#)
 - cursors, X1, X2, Y1, Y2, [633](#)
 - custom time base reference, [1473](#)
 - custom time base reference location, [1474](#)
 - cycle measured, [687](#), [710](#)
 - cycle time, [718](#)
 - cycles analyzed, number of, [910](#), [911](#)
- ## D
- D- USB signal source, [1140](#)
 - D+ USB signal source, [1141](#)
 - damping factor (second-order PLL), clock recovery, [382](#)
 - data, [1034](#), [1036](#), [1563](#)
 - data (waveform) maximum length, [978](#)
 - data 2, [1037](#)
 - data acquisition types, [1556](#)
 - data conversion, [1557](#)
 - data format for transfer, [1558](#)
 - data length, USB search, [1260](#)
 - data length, USB trigger, [1148](#)
 - data output order, [1561](#)
 - data pattern length, [1021](#), [1054](#)
 - data pattern, ARINC 429 trigger, [998](#), [1207](#)
 - data pattern, CAN trigger, [1019](#)
 - data PID, USB search, [1265](#)
 - data PID, USB trigger, [1154](#)
 - data point index, [1587](#)
 - data points, [297](#)
 - data record, measurement, [1568](#)
 - data record, raw acquisition, [1568](#)
 - data required to fill time buckets, [287](#)
 - data source, setup and hold trigger, [1538](#)
 - data structures, status reporting, [1672](#)
 - Data TIE jitter measurement, [752](#)
 - data value, USB search, [1259](#)
 - data value, USB trigger, [1147](#)
 - data, saving and recalling, [409](#)
 - date, calibration, [325](#)
 - date, system, [1425](#)
 - DC coupling for edge trigger, [1506](#)
 - DC input coupling for specified channel, [350](#)
 - DC offset correction for integrate input, [556](#)
 - DC output mode, waveform generator, [1605](#)
 - DC RMS measured on waveform, [763](#)
 - DC waveform generator output, [1610](#)
 - DDE (Device Dependent Error) status bit, [238](#), [240](#)
 - decision chart, status reporting, [1676](#)
 - default conditions, [247](#), [1445](#)
 - define channel labels, [354](#)
 - define delay measurement parameters, [685](#)
 - define glitch trigger, [1517](#)
 - define measurement source, [733](#)
 - define trigger, [1518](#), [1526](#), [1527](#), [1529](#)
 - defined as, [227](#)
 - definite-length block query response, [86](#)
 - definite-length block response data, [228](#)
 - degauss probe, [358](#)
 - delay bits, SPI MISO stream, [1107](#)
 - delay measured to calculate phase, [719](#)
 - delay measurement, [683](#)
 - delay measurement parameters, define, [685](#)
 - delay measurements, [743](#)
 - delay time maximum, Edge Then Edge trigger, [1501](#)
 - delay time minimum, Edge Then Edge trigger, [1502](#)
 - delay time, Edge Then Edge trigger, [1660](#)
 - DElay trigger commands, [1497](#)
 - delay, how autoscale affects, [265](#)
 - delayed time base, [1472](#)
 - delayed window horizontal scale, [1478](#)
 - delete mask, [810](#)
 - delta X cursor, [633](#)
 - delta Y cursor, [633](#)
 - deskew for power measurements, [863](#)
 - destination, remote command logging, [1451](#)
 - detection type, FFT (Magnitude), [473](#)
 - Device Clear to abort a sequential (blocking) command, [92](#)
 - device-defined error queue clear, [236](#)
 - dialog boxes,, [427](#)
 - dialog boxes, close all open, [426](#)
 - differences from 3000G X-Series oscilloscope commands, [55](#)
 - differential D+ and D- USB signal source, [1142](#)
 - differential probe heads, [365](#)
 - differential signal type, [371](#)
 - differentiate math function, [289](#), [561](#), [1577](#)
 - digital channel commands, [396](#), [400](#), [401](#), [404](#)
 - digital channel data, [1559](#)
 - digital channel labels, [439](#)
 - digital channel source for glitch trigger, [1519](#)
 - digital channel, ascending display order, [397](#)
 - digital channel, custom display order, [399](#)
 - digital channel, descending display order, [398](#)
 - digital channel, setup information, [402](#)
 - digital channels, activity and logic levels on, [262](#)
 - digital channels, groups of, [829](#), [831](#), [835](#)
 - digital channels, nibble threshold, [832](#)
 - digital pod, stop displaying, [270](#)
 - digital reset conditions, [248](#), [1446](#)
 - DIGital<d> commands, [395](#)
 - digitize channels, [271](#)
 - DIGitize command, [78](#), [83](#), [1556](#)
 - Digitizer mode, [290](#), [299](#), [309](#)
 - digits, [228](#)
 - disable anti-alias mode, [289](#)
 - disable front panel, [1436](#)
 - disabling channel display, [351](#)
 - disabling status register bits, [237](#), [251](#)
 - discontinued and obsolete commands, [1647](#)
 - display annotation, [410](#)
 - display annotation background, [411](#)
 - display annotation color, [412](#)
 - display annotation text, [416](#)
 - display channel labels, [438](#)
 - display clear, [420](#)
 - DISPlay commands, [405](#)
 - display commands introduction, [409](#)
 - display date, [1425](#)
 - display factors scaling, [356](#)
 - display for channels, [351](#)
 - display frequency span, [487](#), [545](#)
 - display measurements, [671](#), [730](#)
 - display mode, FFT function, [475](#)
 - display persistence, [441](#)
 - display reference, [1469](#)
 - display reference waveforms, [1640](#)
 - display reset conditions, [248](#), [1446](#)
 - display serial number, [276](#)
 - display transparent, [449](#)
 - display vectors, [450](#)
 - display, FFT function, [474](#)
 - display, lister, [615](#)
 - display, measurement statistics on/off, [707](#), [737](#)
 - display, oscilloscope, [400](#), [441](#), [533](#), [831](#), [1427](#)
 - display, serial decode bus, [986](#)
 - displaying a baseline, [1496](#)
 - displaying unsynchronized signal, [1496](#)

divide math function, 561
 DLC value in CAN FD trigger, 1020
 DNS IP, 72
 domain, 72
 duplicate mnemonics, 1714
 duration, 1526, 1527, 1529
 duration for glitch trigger, 1513, 1514, 1518
 duration of power analysis, 912, 913, 914, 915, 916, 917
 duration qualifier, trigger, 1526, 1527
 duration triggering, 1480
 duty cycle measurement, 79, 671, 687, 710
 Duty Cycle, negative, power modulation analysis, 885
 Duty Cycle, positive, power modulation analysis, 885
 DVM commands, 453
 DVM displayed value, 455
 DVM enable/disable, 456
 DVM input source, 458
 DVM mode, 457

E

ECL threshold voltage for digital channels, 404
 edge coupling, 1506
 edge fall time, 688
 edge preshoot measured, 721
 edge rise time, 728
 EDGE SEARCh commands, 1179
 edge search slope, 1180
 edge search source, 1181
 edge slew rate, 731
 edge slope, 1509
 edge source, 1510
 edge string for OR'ed edge trigger, 1521
 EDGE trigger commands, 1505
 edge triggering, 1480
 edges (activity) on digital channels, 262
 edges, measure all, 606
 efficiency, 778
 efficiency power analysis, apply, 864
 elapsed time in mask test, 807
 ellipsis, 228
 enable channel labels, 438
 enabling channel display, 351
 enabling status register bits, 237, 251
 end of string (EOS) terminator, 1710
 end of text (EOT) terminator, 1710
 end or identify (EOI), 1710
 endpoint value, USB search, 1261
 endpoint value, USB trigger, 1149
 energy loss, 779
 envelope math function, 563
 EOI (end or identify), 1710
 EOS (end of string) terminator, 1710
 EOT (end of text) terminator, 1710
 erase data, 420

erase measurements, 1657
 error messages, 1432, 1661
 error number, 1432
 error queue, 1432
 error, measurement, 671
 ESB (Event Status Bit), 252, 254
 ESE (Standard Event Status Enable Register), 237
 ESR (Standard Event Status Register), 239
 ET value, USB search, 1262
 ET value, USB trigger, 1150
 event status conditions occurred, 254
 Event Status Enable Register (ESE), 237
 Event Status Register (ESR), 239, 280
 example code, :ACQure:COMplete, 287
 example code, :ACQure:SEGmented, 304
 example code, :ACQure:TYPE, 312
 example code, :AUToscale, 266
 example code, :CHANnel<n>:LABel, 354
 example code, :CHANnel<n>:PROBe, 356
 example code, :CHANnel<n>:RANGe, 374
 example code, :DIGitize, 272
 example code, :DISPlay:DATA, 425
 example code, :DISPlay:LABel, 438
 example code, :MEASure:PERiod, 734
 example code, :MEASure:RESults, 724
 example code, :MEASure:TEDGe, 743
 example code, :MTESt, 795
 example code, :POD<n>:NIBBle<n>:THReshold, 833
 example code, :POD<n>:THReshold, 835
 example code, :RUN:/STOP, 275
 example code, :SYSTem:SETup, 1457
 example code, :TIMebase:DElay, 1659
 example code, :TIMebase:MODE, 1472
 example code, :TIMebase:RANGe, 1470
 example code, :TIMebase:REference, 1473
 example code, :TRIGger:MODE, 1494
 example code, :TRIGger:SLOPe, 1509
 example code, :TRIGger:SOURce, 1510
 example code, :VIEW and :BLANK, 281
 example code, :WAVEform, 1578
 example code, :WAVEform:DATA, 1563
 example code, :WAVEform:POINts, 1567
 example code, :WAVEform:PREamble, 1571
 example code, :WAVEform:SEGmented, 304
 example code, :WGEN:ARbitrary:DATA, 1597
 example code, *RST, 249
 example programs, 6, 1717
 examples on the website, 1717
 excursion delta for FFT peak search, 1190
 EXE (Execution Error) status bit, 238, 240
 execution error detected in Standard Event Status Register, 240
 explicit clock recovery, 382
 exponential fall waveform generator output, 1611
 exponential math function, 562
 exponential notation, 227

exponential rise waveform generator output, 1610
 external glitch trigger source, 1519
 external offset, probe, 362
 external range, 462
 External Scaling, enable/disable, 360
 External Scaling, gain, 361
 external trigger, 459, 461, 1510
 EXTERNAL trigger commands, 459
 EXTERNAL trigger level, 1507
 external trigger probe attenuation factor, 461
 EXTERNAL trigger source, 1510
 external trigger units, 463

F

factory reset, 1448
 fail count, current harmonics analysis, 870
 fail/pass status (overall) for current harmonics analysis, 877
 failed waveforms in mask test, 805
 failure, self test, 257
 fall time measurement, 671, 688
 Fall Time, power modulation analysis, 885
 falling edge count measurement, 711
 falling pulse count measurement, 713
 FAST data, SENT, 1581
 Fast Fourier Transform (FFT)
 functions, 470, 487, 490, 535, 545, 548, 561
 FF values in waveform data, 1563
 FFT (Fast Fourier Transform) functions, 470, 487, 490, 535, 545, 548, 561
 FFT (Fast Fourier Transform)
 operation, 1577
 FFT (Magnitude) detection type, 473
 FFT bin size, 469
 FFT bin size/RBW/sample rate readout, 544
 FFT detector decimation type, 537
 FFT detector points, 536
 FFT function display, 474
 FFT function display mode, 475
 FFT function, source input, 486
 FFT math function, 289
 FFT Max enable/disable, 495
 FFT Max vertical offset, 492
 FFT Max vertical scale, 494
 FFT Max, rest number of waveforms evaluated, 493
 FFT resolution bandwidth, 482
 FFT resolution display, 483
 FFT sample rate, 488
 FFT vertical units, 489, 547
 FFTPhase (Fast Fourier Transform) functions, 561
 fifty ohm impedance, disable setting, 1449
 filename for recall, 946, 1606
 filename for save, 958
 filter for frequency reject, 1508

filter for high frequency reject, 1484
 filter for noise reject, 1495
 filter used to limit bandwidth, 349, 460
 filters to Fast Fourier Transforms, 490, 548
 filters, math, 562
 fine horizontal adjustment (vernier), 1475
 fine vertical adjustment (vernier), 377
 finish pending device operations, 243
 first point displayed, 1587
 first-order PLL clock recovery, 382
 FLATop window for amplitude measurements, 490, 548, 604
 FM modulation type, waveform generator, 1623
 force trigger, 1483
 format, 1565, 1570
 format (word), ARINC 429, 993
 format for block data, 242
 format for image, 961
 format for waveform data, 976
 FormattedIO488 object, 81
 formulas for data conversion, 1557
 frame, 1110
 frame value, USB search, 1263
 frame value, USB trigger, 1151
 framing, 1108
 FRANalysis commands, 497
 frequency deviation, waveform generator FM modulation, 1618
 frequency measurement, 79, 671, 693
 frequency measurements with X cursors, 643
 frequency mode, Control Loop Response (Bode) power analysis, 852
 frequency mode, Power Supply Rejection Ratio (PSRR) power analysis, 898
 frequency resolution, 490, 548
 frequency response analysis, data, 500
 frequency response analysis, enable, 501
 frequency response analysis, run, 507
 frequency response analysis, single frequency, 503
 frequency response analysis, sweep points, 510
 frequency response analysis, sweep start frequency, 504
 frequency response analysis, sweep stop frequency, 505
 frequency span of display, 487, 545
 Frequency, power modulation analysis, 885
 front panel mode, 1496
 front panel Single key, 277
 front panel Stop key, 279
 front-panel lock, 1436
 FT commands, 465
 full screen waveforms, 432
 full-scale horizontal time, 1470, 1477
 full-scale vertical axis defined, 481, 565
 function, 470, 487, 490, 533, 535, 545, 548, 560, 561, 565, 566, 567
 function auto setup Y scale and offset, 524
 FUNCTION commands, 517

function label, 557
 function memory, 278
 function scale and offset automatic setup, 525
 function, first source input, 569
 function, second source input, 571
 function, waveform generator, 1607
 functions, 1577

G

gain data, including in control loop response results, 859
 gain data, including in FRA results, 511
 gain data, including in PSRR results, 903
 gain for Ax + B math operation, 558
 gated measurement window, 765
 gateway IP, 72
 gating, FFT math function, 478, 540
 gating, math function, 553
 gaussian pulse waveform generator output, 1611
 general SBUS<n> commands, 985
 general SEARCh commands, 1174
 general status commands, 1286
 general trigger commands, 1481
 glitch duration, 1518
 glitch qualifier, 1517
 GLITCh SEARCh commands, 1182
 glitch search, greater than value, 1183
 glitch search, less than value, 1184
 glitch search, polarity, 1185
 glitch search, qualifier, 1186
 glitch search, range, 1187
 glitch search, source, 1188
 glitch source, 1519
 GLITCh trigger commands, 1511
 glitch trigger duration, 1513
 glitch trigger polarity, 1516
 glitch trigger source, 1513
 graticule axis labels, 428
 graticule axis labels, dual, 429
 graticule axis labels, within the grid, 430
 graticule intensity, 433
 greater than qualifier, 1517
 greater than time, 1513, 1518, 1526, 1529
 greater than value for glitch search, 1183
 grid axis labels, 428
 grid axis labels, dual, 429
 grid axis labels, within the grid, 430
 grid intensity, 433
 grid layout, 434
 grid, assign waveform source to grid, 435
 grid, list of waveform sources in, 436
 grids, number of, 431
 groups of digital channels, 829, 831, 835

H

handshake PID, USB search, 1266
 handshake PID, USB trigger, 1155
 HANning window for frequency resolution, 490, 548, 604
 hardcopy factors, 960
 Hardware Operation Register, reading a Condition register bit, 1319
 Hardware Operation Register, reading and clearing an Event register bit, 1323
 Hardware Operation Register, reading and clearing the Event register, 1328
 Hardware Operation Register, reading the Condition register, 1324
 Hardware Operation Register, setting and reading a Negative Transition filter bit, 1321
 Hardware Operation Register, setting and reading a Positive Transition filter bit, 1322
 Hardware Operation Register, setting and reading an Enable register bit, 1320
 Hardware Operation Register, setting and reading the Enable register, 1325
 Hardware Operation Register, setting and reading the Negative Transition filter, 1326
 Hardware Operation Register, setting and reading the Positive Transition filter, 1327
 HARDware status commands, 1316
 HCOPy commands, 573
 head type, probe, 365
 header, 1709
 high pass filter math function, 562
 high trigger level, 1492
 high-frequency reject filter, 1484, 1508
 high-level voltage, waveform generator, 1633
 high-pass filter cutoff frequency, 551
 histogram axis, 580
 histogram commands, 577
 histogram display, 581
 histogram measurement, 582
 histogram mode, 583
 histogram reset, 584
 histogram size, 585
 histogram statistics, saving, 969
 histogram type, 586
 histogram window, bottom limit, 587
 histogram window, left limit, 588
 histogram window, right limit, 589
 histogram window, source, 590
 histogram window, top limit, 591
 histogram, bin width, 694
 histogram, delta value between max and min, 705
 histogram, maximum value, 699
 histogram, mean value, 700
 histogram, median value, 701
 histogram, minimum value, 702

histogram, mode value, [703](#)
 histogram, number of hits in max bin, [704](#)
 histogram, number of total hits, [695](#)
 histogram, pct of hits within +/- 1 std dev, [696](#)
 histogram, pct of hits within +/- 2 std dev, [697](#)
 histogram, pct of hits within +/- 3 std dev, [698](#)
 histogram, standard deviation, [706](#)
 hold time, setup and hold trigger, [1539](#)
 hold until operation complete, [243](#)
 holdoff time, [1485](#)
 holes in waveform data, [1563](#)
 horizontal adjustment, fine (vernier), [1475](#)
 horizontal histogram, [580](#)
 horizontal position, [1476](#)
 horizontal scale, [1471](#), [1478](#)
 horizontal scaling, [1570](#)
 horizontal time, [1470](#), [1477](#)
 Host ID of oscilloscope, [1426](#), [1435](#)
 Host Name, [72](#)
 hostname, [72](#)
 hub address value, USB search, [1264](#)
 hub address value, USB trigger, [1152](#)

I

ID filter for CAN trigger, [1018](#)
 id mode, [1024](#)
 ID pattern, CAN trigger, [1023](#)
 identification number, [241](#)
 identification of options, [244](#)
 identifier, LIN, [1051](#)
 idle state, SENT bus, [1075](#)
 idle until operation complete, [243](#)
 IDN (Identification Number), [241](#)
 IEC 61000-3-2 standard for current harmonics analysis, [876](#)
 IEEE 488.2 standard, [235](#)
 IIC address, [1035](#)
 IIC clock, [1033](#)
 IIC data, [1034](#), [1036](#)
 IIC data 2, [1037](#)
 IIC SEARCh commands, [1220](#)
 IIC serial decode address field size, [1032](#)
 IIC serial search, address, [1223](#)
 IIC serial search, data, [1224](#)
 IIC serial search, data2, [1225](#)
 IIC serial search, mode, [1221](#)
 IIC serial search, qualifier, [1226](#)
 IIC trigger commands, [1031](#)
 IIC trigger qualifier, [1038](#)
 IIC trigger type, [1039](#)
 IIC triggering, [984](#)
 image format, [961](#)
 image memory, [278](#)
 image, save, [959](#)
 impedance, [352](#)
 infinity representation, [1716](#)

initial load current, transient response analysis, [938](#)
 initialization, [78](#), [81](#)
 initialize, [247](#), [1445](#)
 initialize label list, [439](#)
 initiate acquisition, [271](#)
 input coupling for channels, [350](#)
 input for integrate, DC offset correction, [556](#)
 input impedance for channels, [352](#)
 input inversion for specified channel, [353](#)
 input power, [781](#)
 input source, Control Loop Response (Bode) power analysis, [857](#)
 input source, Power Supply Rejection Ratio (PSRR) power analysis, [901](#)
 inrush current, [785](#)
 inrush current analysis, [879](#), [880](#), [881](#)
 inrush current expected, [918](#)
 insert label, [354](#)
 installed options identified, [244](#)
 instruction header, [1709](#)
 Instrument IO, [89](#)
 instrument model number, [273](#)
 instrument number, [241](#)
 instrument options identified, [244](#)
 instrument requests service, [254](#)
 instrument serial number, [276](#)
 instrument status, [88](#)
 instrument type, [241](#)
 integrate DC offset correction, [556](#)
 integrate math function, [561](#), [1577](#)
 Integrate math function, Initial Condition, [555](#)
 intensity color grade scheme, [338](#)
 intensity, waveform, [437](#)
 internal low-pass filter, [349](#), [460](#)
 introduction to :ACQuire commands, [284](#)
 introduction to :BUS<n> commands, [314](#)
 introduction to :CALibrate commands, [324](#)
 introduction to :CHANnel<n> commands, [348](#)
 introduction to :COUNter commands, [385](#)
 introduction to :DIGital<d> commands, [396](#)
 introduction to :DISPlay commands, [409](#)
 introduction to :EXTernal commands, [459](#)
 introduction to :FFT commands, [467](#)
 introduction to :FRANalysis commands, [498](#)
 introduction to :FUNCTion commands, [522](#)
 introduction to :LISTer commands, [613](#)
 introduction to :LTEST commands, [618](#)
 introduction to :MARKer commands, [633](#)
 introduction to :MEASure commands, [671](#)
 introduction to :PBUS commands, [983](#)
 introduction to :POD<n> commands, [829](#)
 introduction to :RECall commands, [942](#)
 introduction to :SAVE commands, [956](#)
 introduction to :SBUS commands, [983](#)
 introduction to :STATus commands, [1283](#)
 introduction to :SYSTem commands, [1424](#)

introduction to :TImebase commands, [1468](#)
 introduction to :TRIGger commands, [1479](#)
 introduction to :WAVEform commands, [1555](#)
 introduction to :WGEN<w> commands, [1594](#)
 introduction to :WMEMory<r> commands, [1637](#)
 introduction to common (*) commands, [235](#)
 introduction to root (:) commands, [261](#)
 inverted masks, bind levels, [821](#)
 inverting input for channels, [353](#)
 IO library, referencing, [80](#)
 IO operation complete, waiting for, [1682](#)
 IP address, [72](#)

J

jitter analysis, enable/disable, [606](#)
 jitter analysis, histogram, [595](#)
 jitter analysis, spectrum, [598](#)
 jitter analysis, spectrum, horizontal range (center), [599](#)
 jitter analysis, spectrum, horizontal range (span), [600](#)
 jitter analysis, spectrum, vertical range, [601](#), [602](#)
 jitter analysis, spectrum, vertical units, [603](#)
 jitter analysis, spectrum, window, [604](#)
 jitter analysis, trend, [608](#)
 jitter analysis, trend smoothing, [609](#)
 jitter analysis, trend smoothing points, [610](#)
 jitter analysis, trend, vertical offset, [611](#)
 jitter analysis, trend, vertical range, [612](#)
 jitter commands, [593](#)
 jitter measurement, [596](#)
 Jitter-Free Trigger disable/enable, [1490](#)

K

key disable, [1436](#)
 key press detected in Standard Event Status Register, [240](#)
 Keysight Interactive IO application, [75](#)
 Keysight IO Control icon, [73](#)
 Keysight IO Libraries Suite, [6](#), [69](#), [80](#), [82](#)
 Keysight IO Libraries Suite, installing, [70](#)
 knob disable, [1436](#)
 known state, [247](#), [1445](#)

L

label command, bus, [320](#)
 label list, [354](#), [439](#)
 label reference waveforms, [1641](#)
 label, ARINC 429 trigger, [997](#), [1001](#), [1205](#)
 label, digital channel, [401](#)

labels, 354, 438, 439
 labels to store calibration information, 326
 labels, specifying, 409
 LAN instrument, 74
 LAN interface, 71, 73
 LAN Settings dialog box, 72
 language for program examples, 77
 leakage into peak spectrum, 490, 548
 learn string, 242, 1457
 least significant byte first, 1561
 left time base reference, 1473
 legal values for channel offset, 355
 legal values for frequency span, 487, 545
 legal values for offset, 560, 566
 length for waveform data, 977
 less than qualifier, 1517
 less than time, 1514, 1518, 1527, 1529
 less than value for glitch search, 1184
 level for trigger voltage, 1507, 1515
 LF coupling, 1506
 license information, 244
 LIN acknowledge, 1045
 LIN baud rate, 1046
 LIN identifier, 1051
 LIN pattern data, 1052
 LIN pattern format, 1055
 LIN SEARCh commands, 1227
 LIN serial decode bus parity bits, 1044
 LIN serial search, data, 1230
 LIN serial search, data format, 1232
 LIN serial search, data length, 1231
 LIN serial search, frame ID, 1228
 LIN serial search, mode, 1229
 LIN source, 1047
 LIN standard, 1048
 LIN symbolic data display, 1043
 LIN symbolic data, recall, 947
 LIN sync break, 1049
 LIN trigger, 1050, 1054
 LIN trigger commands, 1041
 LIN triggering, 984
 line frequency setting for current harmonics analysis, 871
 line glitch trigger source, 1519
 line terminator, 227
 LINE trigger level, 1507
 LINE trigger source, 1510
 Linear units for FFT (Magnitude), 547
 list of channel labels, 439
 LIsTer commands, 613
 lister display, 615
 lister time reference, 616
 ln math function, 562
 local lockout, 1436
 Local Operation Register, reading a Condition register bit, 1332
 Local Operation Register, reading and clearing an Event register bit, 1336
 Local Operation Register, reading and clearing the Event register, 1341
 Local Operation Register, reading the Condition register, 1337

Local Operation Register, setting and reading a Negative Transition filter bit, 1334
 Local Operation Register, setting and reading a Positive Transition filter bit, 1335
 Local Operation Register, setting and reading an Enable register bit, 1333
 Local Operation Register, setting and reading the Enable register, 1338
 Local Operation Register, setting and reading the Negative Transition filter, 1339
 Local Operation Register, setting and reading the Positive Transition filter, 1340
 LOCal status commands, 1329
 location, custom time base reference location, 1474
 lock, 1436
 lock mask to signal, 812
 lock owner session, 1437
 lock release, 1438
 lock request, 1439
 lock, analog channel protection, 1449
 lockout message, 1436
 log file name, remote command logging, 1450, 1453
 log math function, 562
 Logarithmic units for FFT (Magnitude), 547
 long form, 1710
 loop bandwidth (PLL), clock recovery, 382
 low pass filter math function, 562
 low trigger level, 1493
 lower threshold, 718
 lowercase characters in commands, 1709
 low-frequency reject filter, 1508
 low-level voltage, waveform generator, 1634
 low-pass filter cutoff frequency, 552
 low-pass filter used to limit bandwidth, 349, 460
 LRN (Learn Device Setup), 242
 lsbfirst, 1561
 LTESr commands, 617

M

M1553 SEARCh commands, 1236
 M1553 trigger commands, 1059
 M1553 trigger type, 1065
 magnify math function, 563
 magnitude of occurrence, 754
 main sweep range, 1476
 main time base, 1659
 main time base mode, 1472
 making measurements, 671
 manual cursor mode, 635
 manufacturer string, 1440, 1441
 MARKer commands, 631
 marker mode, 646

marker position, 648
 markers track measurements, 730
 markers, command overview, 633
 markers, mode, 635
 markers, X delta, 634, 642
 markers, X1 position, 637
 markers, X1Y1 source, 638
 markers, X2 position, 640
 markers, X2Y2 source, 641
 markers, Y delta, 649
 markers, Y1 position, 646
 markers, Y2 position, 648
 mask, 237, 251
 mask command, bus, 321
 mask statistics, reset, 806
 mask statistics, saving, 970
 mask test commands, 793
 Mask Test Operation Register, reading a Condition register bit, 1345
 Mask Test Operation Register, reading and clearing an Event register bit, 1349
 Mask Test Operation Register, reading and clearing the Event register, 1354
 Mask Test Operation Register, reading the Condition register, 1350
 Mask Test Operation Register, setting and reading a Negative Transition filter bit, 1347
 Mask Test Operation Register, setting and reading a Positive Transition filter bit, 1348
 Mask Test Operation Register, setting and reading an Enable register bit, 1346
 Mask Test Operation Register, setting and reading the Enable register, 1351
 Mask Test Operation Register, setting and reading the Negative Transition filter, 1352
 Mask Test Operation Register, setting and reading the Positive Transition filter, 1353
 mask test run mode, 813
 mask test termination conditions, 813
 mask test, all channels, 799
 mask test, enable/disable, 811
 mask, delete, 810
 mask, get as binary block data, 809
 mask, load from binary block data, 809
 mask, lock to signal, 812
 mask, recall, 948
 mask, save, 962, 963
 masks, bind levels, 821
 master summary status bit, 254
 math filters, 562
 math function auto setup Y scale and offset, 524
 math function label, 557
 math function scale and offset automatic setup, 525
 math function, stop displaying, 270
 math operators, 561
 math source in all color grade analysis, 340

- math transforms, 561
 - math visualizations, 563
 - MAV (Message Available), 236, 252, 254
 - max hold math function, 563
 - maximum duration, 1514, 1526, 1527
 - maximum math function, 563
 - maximum number of peaks for FFT peak search, 1191
 - maximum position, 1469
 - maximum random trigger holdoff time, 1486
 - maximum range for zoomed window, 1477
 - maximum scale for zoomed window, 1478
 - maximum vertical value measurement, 759
 - maximum vertical value, time of, 766
 - maximum waveform data length, 978
 - measure all edges, 606
 - MEASure commands, 653
 - measure mask test failures, 814
 - measure overshoot, 715
 - measure period, 718
 - measure phase between channels, 719
 - MEASure power commands, 769
 - measure preshoot, 721
 - measure value at a specified time, 768
 - measure value at top of waveform, 764
 - measurement error, 671
 - measurement limit test, copy all limits from results, 620
 - measurement limit test, copy limit from results, 619
 - measurement limit test, enable, 622
 - measurement limit test, enable/disable, 628
 - measurement limit test, fail condition, 623
 - measurement limit test, lower limit, 624, 629
 - measurement limit test, margin when copying from results, 621
 - measurement limit test, measurement source, 625
 - measurement limit test, results, 626
 - measurement limit test, stop on failure, 627
 - measurement results, saving, 971
 - measurement setup, 671, 733
 - measurement source, 733
 - measurement statistics results, 724
 - measurement statistics, display on/off, 707, 737
 - measurement thresholds, absolute values, 746
 - measurement thresholds, mode, 747
 - measurement thresholds, percent values, 748
 - measurement thresholds, set up standard, 749
 - measurement trend math function, 563
 - measurement window, 765
 - measurements, AC RMS, 763
 - measurements, area, 673
 - measurements, average value, 757
 - measurements, base value, 758
 - measurements, built-in, 79
 - measurements, burst width, 675
 - measurements, clear, 676, 1657
 - measurements, command overview, 671
 - measurements, counter, 677
 - measurements, DC RMS, 763
 - measurements, fall time, 688
 - measurements, falling edge count, 711
 - measurements, falling pulse count, 713
 - measurements, frequency, 693
 - measurements, how autoscale affects, 265
 - measurements, maximum vertical value, 759
 - measurements, maximum vertical value, time of, 766
 - measurements, minimum vertical value, 760
 - measurements, minimum vertical value, time of, 767
 - measurements, negative duty cycle, 710
 - measurements, overshoot, 715
 - measurements, period, 718
 - measurements, phase, 719
 - measurements, positive duty cycle, 687
 - measurements, preshoot, 721
 - measurements, pulse width, negative, 714
 - measurements, pulse width, positive, 722
 - measurements, ratio of AC RMS values, 762
 - measurements, rise time, 728
 - measurements, rising edge count, 717
 - measurements, rising pulse count, 720
 - measurements, show, 730
 - measurements, slew rate, 731
 - measurements, snapshot all, 723
 - measurements, source channel, 733
 - measurements, standard deviation, 729
 - measurements, time between trigger and edge, 742
 - measurements, time between trigger and vertical value, 754
 - measurements, vertical amplitude, 756
 - measurements, vertical peak-to-peak, 761
 - memory depth, 3
 - memory depth (User) setting, user interface, 294
 - memory depth, setting, 297
 - memory setup, 1457
 - message available bit, 254
 - message available bit clear, 236
 - message decode/triggering format, SENT, 1073
 - message displayed, 254
 - message error, 1661
 - message queue, 1674
 - message, CAN symbolic search, 1217
 - message, CAN symbolic trigger, 1025
 - message, clear from display, 440
 - message, LIN symbolic search, 1233
 - message, LIN symbolic trigger, 1056
 - messages ready, 254
 - method of clock recovery, 382
 - midpoint of thresholds, 718
 - MIL-STD-1553 serial decode base, 1061
 - MIL-STD-1553 serial search, data, 1238
 - MIL-STD-1553 serial search, mode, 1237
 - MIL-STD-1553 serial search, Remote Terminal Address, 1239
 - MIL-STD-1553 triggering, 984
 - min hold math function, 563
 - minimum duration, 1513, 1526, 1527, 1529
 - minimum math function, 563
 - minimum random trigger holdoff time, 1487
 - minimum vertical value measurement, 760
 - minimum vertical value, time of, 767
 - MISO data pattern width, 1114
 - MISO data pattern, SPI trigger, 1113
 - MISO data source, SPI trigger, 1111
 - MISO data, SPI, 1581
 - mnemonics, duplicate, 1714
 - mode, 635, 1472
 - mode, serial decode, 987
 - model number, 241, 273
 - models, oscilloscope, 3
 - modes for triggering, 1494
 - modulating signal frequency, waveform generator, 1617, 1619
 - modulation (waveform generator), enabling/disabling, 1622
 - modulation analysis, 883
 - modulation analysis source (voltage or current), 884
 - modulation analysis, type of, 885
 - modulation type, waveform generator, 1623
 - MOSI data pattern width, 1116
 - MOSI data pattern, SPI trigger, 1115
 - MOSI data source, SPI trigger, 1112, 1658
 - most significant byte first, 1561
 - msbfirst, 1561
 - MSG (Message), 252, 254
 - MSS (Master Summary Status), 254
 - MTESt commands, 793
 - MTESt status commands, 1342
 - multi-channel waveform data, save, 964
 - multiple commands, 1714
 - multiple queries, 87
 - multiplier value for SENT Fast Channel Signal, 1082
 - multiply math function, 561, 1577
- ## N
- N275xA InfiniiMode probe mode, 367
 - N275xA InfiniiMode probe quick-action button, 357
 - N8900A Infiniium Offline oscilloscope analysis software, 964
 - name channels, 354
 - name list, 439

natural logarithm math function, [562](#)
 negative glitch trigger polarity, [1516](#)
 negative pulse width, [714](#)
 negative pulse width measurement, [79](#)
 negative pulse width, power modulation analysis, [885](#)
 negative slope, [1105](#), [1509](#)
 Network Time Protocol, setting oscilloscope's clock, [1462](#)
 new line (NL) terminator, [227](#), [1710](#)
 new load current, transient response analysis, [939](#)
 nibble order for SENT Fast Channel Signal, [1086](#)
 nibble threshold, digital channels, [832](#)
 NL (new line) terminator, [227](#), [1710](#)
 noise floor, [932](#), [935](#)
 noise reject filter, [1495](#)
 noise waveform generator output, [1610](#)
 noise, adding to waveform generator output, [1621](#)
 nominal data rate, clock recovery, [382](#)
 non-core commands, [1708](#)
 non-volatile memory, label list, [320](#), [401](#), [439](#)
 normal acquisition type, [284](#), [1556](#)
 normal trigger sweep mode, [1479](#)
 notices, [2](#)
 N-Period jitter measurement, [712](#)
 N-Period to N-Period jitter measurement, [708](#)
 NR1 number format, [227](#)
 NR3 number format, [227](#)
 null offset, [932](#)
 NULL string, [1427](#)
 number format, [227](#)
 number of nibbles, SENT message, [1076](#)
 number of points, [297](#), [1566](#), [1568](#)
 number of time buckets, [1566](#), [1568](#)
 numeric variables, [86](#)
 numeric variables, reading query results into multiple, [87](#)
 nwidth, [714](#)

O

obsolete and discontinued commands, [1647](#)
 obsolete commands, [1708](#)
 Occupied Bandwidth, FFT analysis measurement, [691](#)
 offset, [523](#)
 offset for Ax + B math operation, [559](#)
 offset value for channel voltage, [355](#)
 offset value for FFT function, [480](#), [484](#)
 offset value for selected function, [560](#), [566](#)
 offset value for SENT Fast Channel Signal, [1084](#)
 offset, waveform generator, [1635](#)
 one values in waveform data, [1563](#)
 OPC (Operation Complete) command, [243](#)

OPC (Operation Complete) status bit, [238](#), [240](#)
 Open method, [81](#)
 operating configuration, [242](#), [1457](#)
 operating state, [250](#)
 operation complete, [243](#)
 OPERation status commands, [1288](#)
 operation status conditions occurred, [254](#)
 Operation Status Register, reading a Condition register bit, [1293](#)
 Operation Status Register, reading and clearing an Event register bit, [1297](#)
 Operation Status Register, reading and clearing the Event register, [1302](#)
 Operation Status Register, reading the Condition register, [1298](#)
 Operation Status Register, setting and reading a Negative Transition filter bit, [1295](#)
 Operation Status Register, setting and reading a Positive Transition filter bit, [1296](#)
 Operation Status Register, setting and reading an Enable register bit, [1294](#)
 Operation Status Register, setting and reading the Enable register, [1299](#)
 Operation Status Register, setting and reading the Negative Transition filter, [1300](#)
 Operation Status Register, setting and reading the Positive Transition filter, [1301](#)
 operations for function, [561](#)
 operators, math, [561](#)
 OPT (Option Identification), [244](#)
 optional syntax terms, [227](#)
 options, [244](#)
 OR trigger commands, [1520](#)
 order of output, [1561](#)
 oscilloscope connection, opening, [81](#)
 oscilloscope connection, verifying, [73](#)
 oscilloscope external trigger, [459](#)
 oscilloscope models, [3](#)
 oscilloscope, connecting, [71](#)
 oscilloscope, initialization, [78](#)
 oscilloscope, operation, [6](#)
 oscilloscope, program structure, [78](#)
 oscilloscope, setting up, [71](#)
 oscilloscope, setup, [82](#)
 output control, waveform generator, [1624](#)
 output load impedance, waveform generator, [512](#), [1627](#)
 output messages ready, [254](#)
 output polarity, waveform generator, [1629](#)
 output power, [784](#)
 output queue, [243](#), [1673](#)
 output queue clear, [236](#)
 output ripple, [790](#)
 output ripple analysis, [908](#)
 output sequence, [1561](#)
 output source, Control Loop Response (Bode) power analysis, [858](#)

output source, Power Supply Rejection Ratio (PSRR) power analysis, [902](#)
 overall pass/fail status for current harmonics analysis, [877](#)
 overlapped commands, [91](#)
 overload, [373](#)
 Overload Operation Register, reading a Condition register bit, [1358](#)
 Overload Operation Register, reading and clearing an Event register bit, [1362](#)
 Overload Operation Register, reading and clearing the Event register, [1367](#)
 Overload Operation Register, reading the Condition register, [1363](#)
 Overload Operation Register, setting and reading a Negative Transition filter bit, [1360](#)
 Overload Operation Register, setting and reading a Positive Transition filter bit, [1361](#)
 Overload Operation Register, setting and reading an Enable register bit, [1359](#)
 Overload Operation Register, setting and reading the Enable register, [1364](#)
 Overload Operation Register, setting and reading the Negative Transition filter, [1365](#)
 Overload Operation Register, setting and reading the Positive Transition filter, [1366](#)
 OVERload status commands, [1355](#)
 overshoot of waveform, [715](#)
 overshoot percent for transient response analysis, [919](#)
 overvoltage, [373](#)

P

parametric measurements, [671](#)
 parity, [1125](#)
 parity bits, LIN serial decode bus, [1044](#)
 parser, [261](#), [1714](#)
 pass, self test, [257](#)
 pass/fail status (overall) for current harmonics analysis, [877](#)
 path information, recall, [949](#)
 path information, save, [966](#)
 pattern, [1035](#), [1036](#), [1037](#)
 pattern data, LIN, [1052](#)
 pattern duration, [1513](#), [1514](#), [1526](#), [1527](#)
 pattern for pattern trigger, [1523](#)
 pattern format, LIN, [1055](#)
 pattern length, [1021](#), [1054](#)
 PATtern trigger commands, [1522](#)
 pattern trigger format, [1525](#)
 pattern trigger qualifier, [1528](#)
 pattern triggering, [1480](#)
 pattern width, [1114](#), [1116](#)
 pause pulse, SENT messages, [1077](#)
 PBUS commands, [983](#)

- peak current, 785
 - peak data, 1557
 - peak detect, 311
 - peak detect acquisition type, 285, 1557
 - PEAK SEARCh commands, 1189
 - peak-peak math function, 563
 - peaks in analysis record, 1444
 - peak-to-peak vertical value measurement, 761
 - pending operations, 243
 - percent of waveform overshoot, 715
 - period measured to calculate phase, 719
 - period measurement, 79, 671, 718
 - Period, power modulation analysis, 885
 - period, waveform generator, 1630
 - persistence data, clear from display, 442
 - persistence, waveform, 409, 441
 - PFAult status commands, 1368
 - phase data, including in control loop response results, 859
 - phase data, including in FRA results, 511
 - phase measured between channels, 719
 - phase measurements, 743
 - phase measurements with X cursors, 643
 - PID check value, USB trigger, 1153
 - PNG format screen image data, 425, 575
 - pod, 829, 831, 834, 835, 1577
 - POD commands, 829
 - POD data format, 1559
 - pod, nibble threshold, 832
 - pod, stop displaying, 270
 - points, 297, 1566, 1568
 - points in waveform data, 1556
 - points per decade, Control Loop Response (Bode) power analysis, 856
 - points per decade, frequency response analysis, 506
 - points per decade, Power Supply Rejection Ratio (PSRR) power analysis, 900
 - points per span, FFT (Magnitude), 472
 - points, setting, 297
 - polarity, 1126
 - polarity for glitch search, 1185
 - polarity for glitch trigger, 1516
 - polarity, runt search, 1195
 - polarity, runt trigger, 1531
 - polling synchronization example, 1695
 - polling synchronization with timeout, 1688
 - polling wait, 1686
 - PON (Power On) status bit, 238, 240
 - port value, USB search, 1269
 - port value, USB trigger, 1158
 - position, 640, 1469, 1476
 - position in zoomed view, 1476
 - positive glitch trigger polarity, 1516
 - positive pulse width, 722
 - positive pulse width measurement, 79
 - positive pulse width, power modulation analysis, 885
 - positive slope, 1105, 1509
 - positive width, 722
 - power analysis, enabling, 866
 - POWer commands, 837
 - power cycle the oscilloscope, 1442
 - power factor, 780
 - power factor for IEC 61000-3-2 Standard Class C, 872
 - power loss, 786
 - Power Operation Register, reading a Condition register bit, 1384
 - Power Operation Register, reading and clearing an Event register bit, 1388
 - Power Operation Register, reading and clearing the Event register, 1393
 - Power Operation Register, reading the Condition register, 1389
 - Power Operation Register, setting and reading a Negative Transition filter bit, 1386
 - Power Operation Register, setting and reading a Positive Transition filter bit, 1387
 - Power Operation Register, setting and reading an Enable register bit, 1385
 - Power Operation Register, setting and reading the Enable register, 1390
 - Power Operation Register, setting and reading the Negative Transition filter, 1391
 - Power Operation Register, setting and reading the Positive Transition filter, 1392
 - power phase angle, 774
 - power quality analysis, 907
 - POWer status commands, 1381
 - power supply rejection ratio (PSRR), 894, 896, 897, 899
 - Power Supply Rejection Ratio (PSRR) power analysis settings, 893
 - preamble data, 1570
 - preamble metadata, 1555
 - precision analysis record, 1568
 - precision analysis record length, 1443
 - present working directory, recall operations, 949
 - present working directory, save operations, 966
 - preset conditions, 1445
 - preshoot measured on waveform, 721
 - previously stored configuration, 246
 - print mask test failures, 815
 - probe, 1507
 - probe attenuation affects channel voltage range, 374
 - probe attenuation factor (external trigger), 461
 - probe attenuation factor for selected channel, 356
 - Probe Fault Register, reading a Condition register bit, 1371
 - Probe Fault Register, reading and clearing an Event register bit, 1375
 - Probe Fault Register, reading and clearing the Event register, 1380
 - Probe Fault Register, reading the Condition register, 1376
 - Probe Fault Register, setting and reading a Negative Transition filter bit, 1373
 - Probe Fault Register, setting and reading a Positive Transition filter bit, 1374
 - Probe Fault Register, setting and reading an Enable register bit, 1372
 - Probe Fault Register, setting and reading the Enable register, 1377
 - Probe Fault Register, setting and reading the Negative Transition filter, 1378
 - Probe Fault Register, setting and reading the Positive Transition filter, 1379
 - probe head type, 365
 - probe ID, 366
 - probe skew value, 369
 - process sigma, mask test run, 818
 - program data, 1710
 - program data syntax rules, 1712
 - program initialization, 78
 - program message, 81, 235
 - program message syntax, 1709
 - program message terminator, 1710
 - program message units, 1710
 - program message units, multiple, 1713
 - program structure, 78
 - programming examples, 6, 1717
 - protection, 373
 - protection lock, 1449
 - protocol decode bus, 983
 - pulse waveform generator output, 1609
 - pulse width, 714, 722
 - pulse width duration trigger, 1513, 1514, 1518
 - pulse width measurement, 79, 671
 - pulse width trigger, 1495
 - pulse width trigger level, 1515
 - pulse width triggering, 1480
 - pulse width, waveform generator, 1613
 - pwidth, 722
 - Python, VISA COM example, 1745
 - Python, VISA example, 1782
 - Python, VISA.NET example, 1801
 - PyVISA package, 1782
- ## Q
- qualifier, 1518
 - qualifier for glitch search, 1186
 - qualifier, runt search, 1196
 - qualifier, runt trigger, 1532
 - qualifier, transition search, 1200
 - qualifier, transition trigger, 1542
 - qualifier, trigger duration, 1526, 1527
 - qualifier, trigger pattern, 1528
 - queries, multiple, 87
 - query error detected in Standard Event Status Register, 240
 - query responses, block data, 86
 - query responses, reading, 85

query results, reading into numeric variables, 86
 query results, reading into string variables, 85
 query return values, 1716
 query setup, 633, 671, 1457
 query subsystem, 314, 396
 querying setup, 348
 querying the subsystem, 1480
 queues, clearing, 1675
 Quick Measure, 723
 quick reference, commands, 93
 quoted ASCII string, 228
 QYE (Query Error) status bit, 238, 240

R

ramp symmetry, waveform generator, 1614
 ramp waveform generator output, 1609
 random trigger holdoff, 1488
 range, 523, 1477
 range for channels, 374
 range for duration trigger, 1529
 range for external trigger, 462
 range for full-scale vertical axis, 481, 565
 range for glitch search, 1187
 range for glitch trigger, 1518
 range for time base, 1470
 range of offset values, 355
 range qualifier, 1517
 ranges, value, 228
 ratio measurements with X cursors, 643
 ratio measurements with Y cursors, 650
 ratio of AC RMS values measured between channels, 762
 Ratio, power modulation analysis, 885
 raw acquisition record, 1568
 RCL (Recall), 246
 Rds (dynamic ON resistance)
 waveform, 931
 Rds(on) power measurement, 787
 Rds(on) value for conduction
 calculation, 933
 reactive power, 788
 read configuration, 242
 ReadIEEEBlock method, 81, 85, 87
 ReadList method, 81, 85
 ReadNumber method, 81, 85
 ReadString method, 81, 85
 real power, 789
 Real Power source in Class D harmonics
 analysis, 873
 real-time acquisition mode, 296
 recall, 246, 942, 1457
 recall arbitrary waveform, 944
 recall CAN symbolic data, 945
 RECall commands, 941
 recall filename, 946, 1606
 recall LIN symbolic data, 947
 recall mask, 948
 recall path information, 949
 recall reference waveform, 951
 recall setup, 950
 recalling and saving data, 409
 recovered clock, displaying, 381
 RECTangular window for transient
 signals, 490, 548, 604
 reference, 523
 reference for time base, 1659
 reference point, FFT Phase, 542
 reference waveform save source, 980
 reference waveform, recall, 951
 reference waveform, save, 981
 reference waveforms, clear, 1639
 reference waveforms, display, 1640
 reference waveforms, label, 1641
 reference waveforms, save to, 1642
 reference waveforms, skew, 1643
 reference waveforms, Y offset, 1644
 reference waveforms, Y range, 1645
 reference waveforms, Y scale, 1646
 reference, lister, 616
 reference, time base, 1473
 registers, 239
 registers, clearing, 1675
 reject filter, 1508
 reject high frequency, 1484
 reject noise, 1495
 relative standard deviation, 741
 remote command logging,
 enable/disable, 1450, 1454
 remote control examples, 1717
 Remote Terminal Address (RTA), M1553
 trigger, 1064
 remote user interface enabled/disabled
 status, 1292
 remove cursor information, 635
 remove labels, 438
 remove message from display, 1427
 reorder channels, 265
 repetitive acquisitions, 275
 report errors, 1432
 report transition, 754
 reporting status, 1669
 reporting the setup, 1480
 request service, 254
 Request-for-OPC flag clear, 236
 reset, 247
 reset conditions, 247
 reset defaults, waveform generator, 1631
 reset mask statistics, 806
 reset measurements, 420
 Resolution Bandwidth, FFT, 543
 resource session object, 81
 ResourceManager object, 81
 restore configurations, 242, 246, 250,
 1457
 restore labels, 438
 restore setup, 246
 results area layout, 444
 results area list, window select, 445
 results area size, 448

results area tabbed, left pane window
 select, 446
 results area tabbed, right pane window
 select, 447
 results area window list, 443
 return values, query, 1716
 returning acquisition type, 311
 returning number of data points, 297
 right time base reference, 1473
 ripple (output) analysis, 908
 ripple, output, 790
 rise time measurement, 671
 rise time of positive edge, 728
 Rise Time, power modulation analysis, 885
 rising edge count measurement, 717
 rising pulse count measurement, 720
 RMS - AC, power modulation analysis, 885
 RMS value measurement, 763
 roll time base mode, 1472
 root (:) commands, 259, 261
 root level commands, 3
 RQL (Request Control) status bit, 238, 240
 RQS (Request Service), 254
 RS-232/UART triggering, 984
 R-sense resistor value, N2825A
 user-defined R-sense head, 368
 RST (Reset), 247
 rules, tree traversal, 1714
 rules, truncation, 1710
 run, 256, 275
 run count, current harmonics analysis, 875
 run mode, mask test, 813
 run state, 274
 running configuration, 250, 1457
 RUNT SEARCh commands, 1194
 runt search polarity, 1195
 runt search qualifier, 1196
 runt search source, 1197
 runt search, pulse time, 1198
 RUNT trigger commands, 1530
 runt trigger polarity, 1531
 runt trigger qualifier, 1532
 runt trigger source, 1533
 runt trigger time, 1534
 Rx source, 1127

S

S and E or U value, USB search, 1271
 S and E or U value, USB trigger, 1160
 sample rate, 3, 307
 sample rate (User) setting, user
 interface, 295
 Sample Rate, FFT, 546
 sample rate, setting, 307
 sampled data points, 1563
 SAV (Save), 250
 save, 250, 956
 save arbitrary waveform, 957
 SAVE commands, 953

- save current harmonics analysis results, 965
- save filename, 958
- save image, 959
- save mask, 962, 963
- save mask test failures, 816
- save path information, 966
- save reference waveform, 981
- save setup, 974
- save to reference waveform location, 1642
- save waveform data, 975
- saving and recalling data, 409
- SBUS A429 commands, 988
- SBUS CAN commands, 1004
- SBUS commands, 983
- SBUS SENT commands, 1066
- SBUS USBPd commands, 1161
- SBUS<n> commands, general, 985
- SC value, USB search, 1270
- SC value, USB trigger, 1159
- scale, 485, 567, 1471, 1478
- scale factors output on hardcopy, 960
- scale for channels, 375
- scale units for channels, 364, 376
- scale units for external trigger, 463
- scaling display factors, 356
- scheme, color grade, 342
- SCPI commands, 89
- SCPI.NET examples, 1818
- scratch measurements, 1657
- screen display of logged remote commands, enable/disable, 1452
- screen image data, 425, 574
- SDI pattern, ARINC 429 trigger, 999, 1208
- SEARCh commands, 1173
- SEARCh commands, A429, 1204
- SEARCh commands, CAN, 1210
- SEARCh commands, EDGE, 1179
- SEARCh commands, general, 1174
- SEARCh commands, GLITCh, 1182
- SEARCh commands, IIC, 1220
- SEARCh commands, LIN, 1227
- SEARCh commands, M1553, 1236
- SEARCh commands, PEAK, 1189
- SEARCh commands, RUNT, 1194
- SEARCh commands, SENT, 1240
- SEARCh commands, SPI, 1245
- SEARCh commands, TRANSition, 1199
- SEARCh commands, UART, 1249
- SEARCh commands, USB, 1254
- SEARCh commands, USBPd, 1272
- search event (found) times, saving, 972
- search for found event, 1176
- search mode, 1177
- search state, 1178
- search, edge slope, 1180
- search, edge source, 1181
- search, USB, 1256
- second-order PLL clock recovery, 382
- seconds per division, 1471
- segmented memory acquisition times, 973
- segmented waveform save option, 979
- segments, analyze, 302
- segments, count of waveform, 1574
- segments, setting number of memory, 303
- segments, setting the index, 304
- segments, time tag, 1575
- select measurement channel, 733
- self-test, 257
- sensitivity of oscilloscope input, 356
- SENT enhanced serial message ID and data lengths, 1100
- SENT fast channel data search value, 1241
- SENT fast channel data trigger setting, 1095
- SENT FAST data, 1581
- SENT input source, 1090
- SENT percent tolerance variation trigger, 1101
- SENT SEARCh commands, 1240
- SENT search mode, 1242
- SENT serial bus commands, 1066
- SENT signal length setting, 1080
- SENT signals display setting, 1079
- SENT slow channel data search value, 1243
- SENT slow channel ID and data trigger data setting, 1096
- SENT slow channel ID search value, 1244
- SENT slow channel ID trigger setting, 1098
- SENT SLOW data, 1581
- SENT trigger mode, 1093
- SENT triggering, 984
- sequential commands, 91
- serial clock, 1033, 1109
- serial data, 1034
- serial decode bus, 983
- serial decode bus display, 986
- serial decode mode, 987
- serial frame, 1110
- serial number, 276
- service request, 254
- Service Request Enable Register (SRE), 251, 252
- set center frequency, 470, 535
- set date, 1425
- set time, 1460
- set up oscilloscope, 71
- setting digital display, 400
- setting digital label, 320, 401
- setting digital threshold, 404
- setting display, 533
- setting external trigger level, 459
- setting impedance for channels, 352
- setting inversion for channels, 353
- setting pod display, 831
- setting pod size, 834
- setting pod threshold, 835
- settings conflict errors, 1292
- setup, 285, 314, 348, 396, 409, 1457
- setup and hold trigger clock source, 1537
- setup and hold trigger data source, 1538
- setup and hold trigger hold time, 1539
- setup and hold trigger setup time, 1540
- setup and hold trigger slope, 1536
- setup configuration, 246, 250, 1457
- setup defaults, 247, 1445
- setup reported, 1480
- setup time, setup and hold trigger, 1540
- setup, recall, 950
- setup, save, 974
- shape of modulation signal, waveform generator, 1620
- SHOLd trigger commands, 1535
- short form, 6, 1710
- show channel labels, 438
- show measurements, 671, 730
- shutdown the oscilloscope, 1459
- SiCL example in C, 1808
- SiCL examples, 1808
- sigma, mask test run, 818
- signal speed, ARINC 429, 996
- signal type, 371
- signal type, ARINC 429, 994
- signal value, CAN symbolic search, 1219
- signal value, CAN symbolic trigger, 1027
- signal value, LIN symbolic search, 1235
- signal value, LIN symbolic trigger, 1058
- signal, CAN symbolic search, 1218
- signal, CAN symbolic trigger, 1026
- signal, LIN symbolic search, 1234
- signal, LIN symbolic trigger, 1057
- signed data, 1558
- simple command headers, 1711
- sine cardinal waveform generator output, 1610
- sine waveform generator output, 1608
- single acquisition, 277
- single frequency, frequency response analysis, 502
- single-ended probe heads, 365
- single-ended signal type, 371
- single-shot DUT, synchronizing with, 1690
- size, 834
- size, digital channels, 403
- skew, 369
- skew reference waveform, 1643
- slew rate of an edge, 731
- slew rate power analysis, 928
- slope, 1105, 1509
- slope not valid in TV trigger mode, 1509
- slope, arming edge, Edge Then Edge trigger, 1499
- slope, setup and hold trigger, 1536
- slope, transition search, 1201
- slope, transition trigger, 1543
- slope, trigger edge, Edge Then Edge trigger, 1503
- SLOW data, SENT, 1581
- smoothing math function, 562
- smoothing math function, number of points, 568
- software version, 241
- source, 733, 995, 1014, 1047
- source (voltage or current) for slew rate analysis, 929

source channel, M1553, 1062
 source for glitch search, 1188
 source for trigger, 1510
 source function for FFT peak search, 1192
 source input for FFT function, 486
 source input for function, first, 569
 source input for function, second, 571
 source, arming edge, Edge Then Edge trigger, 1500
 source, automask, 801
 source, mask test, 826
 source, runt search, 1197
 source, runt trigger, 1533
 source, save reference waveform, 980
 source, transition trigger, 1202, 1544
 source, trigger edge, Edge Then Edge trigger, 1504
 source, waveform, 1577
 span, 561
 span of frequency on display, 487, 545
 special PID, USB search, 1267
 special PID, USB trigger, 1156
 specify measurement, 733
 speed of ARINC 429 signal, 996
 speed, USB decode, 1143
 SPI, 1105
 SPI clock timeout, 1106
 SPI decode bit order, 1104
 SPI decode word width, 1118
 SPI MISO data, 1581
 SPI SEARCh commands, 1245
 SPI serial search, data, 1247
 SPI serial search, data width, 1248
 SPI serial search, mode, 1246
 SPI trigger, 1108, 1114, 1116
 SPI trigger clock, 1109
 SPI trigger commands, 1102
 SPI trigger frame, 1110
 SPI trigger MISO data pattern, 1113
 SPI trigger MOSI data pattern, 1115
 SPI trigger type, 1117
 SPI trigger, MISO data source, 1111
 SPI trigger, MOSI data source, 1112, 1658
 SPI triggering, 984
 square math function, 562
 square root math function, 562
 square wave duty cycle, waveform generator, 1615
 square waveform generator output, 1608
 SRE (Service Request Enable Register), 251, 252
 SSM pattern, ARINC 429 trigger, 1000, 1209
 standard deviation measured on waveform, 729
 Standard Event Status Enable Register (ESE), 237
 Standard Event Status Register (ESR), 239
 standard, LIN, 1048
 start acquisition, 256, 271, 275, 277
 start frequency, FFT math function, 476, 538

start measurement, 671
 start time, 1518
 starting bit for SENT Fast Channel Signal, 1088
 STAT OPERAtion OVERload commands, 1355
 state of instrument, 242, 1457
 state, run, 274
 statistics increment, 738
 statistics reset, 740
 statistics results, 724
 statistics, max count, 739
 statistics, relative standard deviation, 741
 statistics, type of, 736
 status, 253, 278, 280
 Status Byte Register (STB), 253, 254
 STATus commands, 1283
 STATus commands, general, 1286
 status data structure clear, 236
 STATus OPERAtion ARM commands, 1303
 STATus OPERAtion commands, 1288
 STATus OPERAtion HARDware commands, 1316
 STATus OPERAtion LOCAL commands, 1329
 STATus OPERAtion MTEST commands, 1342
 STATus OPERAtion OVERload PFAult commands, 1368
 STATus OPERAtion POWER commands, 1381
 status register preset, 1287
 status registers, 88
 status reporting, 1669
 STATus TRIGger commands, 1394
 STATus USER commands, 1407
 STB (Status Byte Register), 253, 254
 steady state output voltage expected, 923, 924, 925
 step size for frequency span, 487, 545
 stop, 271, 279
 stop acquisition, 279
 stop displaying channel, 270
 stop displaying math function, 270
 stop displaying pod, 270
 stop frequency, FFT math function, 477, 539
 stop on mask test failure, 817
 stop time, 1518
 store instrument setup, 242, 250
 store setup, 250
 storing calibration information, 326
 string variables, 85
 string variables, reading multiple query results into, 87
 string variables, reading query results into multiple, 87
 string, quoted ASCII, 228
 subnet mask, 72
 subsources, waveform source, 1581
 subsystem commands, 3, 1714
 subtract math function, 561, 1577
 sweep mode, trigger, 1479, 1496

sweep speed set to fast to measure fall time, 688
 sweep speed set to fast to measure rise time, 728
 switch disable, 1436
 switching level, current, 932
 switching level, voltage, 935
 switching loss per cycle, 776
 switching loss power analysis, 930
 sync break, LIN, 1049
 syntax elements, 227
 syntax rules, program data, 1712
 syntax, optional terms, 227
 syntax, program message, 1709
 SYSTem commands, 1421
 system commands, 1425, 1427, 1432, 1436, 1457, 1460
 system commands introduction, 1424

T

table of current harmonics results, 869
 telnet ports 5024 and 5025, 1563
 Telnet sockets, 89
 temperature color grade scheme, 338
 temporary message, 1427
 TER (Trigger Event Register), 280
 termination conditions, mask test, 813
 test sigma, mask test run, 818
 test, self, 257
 text, writing to display, 1427
 THD (total harmonics distortion), 878
 theme, color grade, 342
 threshold, 404, 835
 threshold for FFT peak search, 1193
 threshold, digital channel nibble, 832
 thresholds used to measure period, 718
 thresholds, how autoscale affects, 265
 time base, 1469, 1470, 1471, 1472, 1659
 time base commands introduction, 1468
 time base reference, 1473
 time base reset conditions, 248, 1446
 time base window, 1476, 1477, 1478
 time buckets, 287, 288
 time delay, 1659
 time delay maximum, Edge Then Edge trigger, 1501
 time delay minimum, Edge Then Edge trigger, 1502
 time delay, Edge Then Edge trigger, 1660
 time difference between data points, 1585
 time duration, 1518, 1526, 1527, 1529
 time holdoff for trigger, 1485
 time interval, 754
 time measurements with X cursors, 643
 time per division, 1470
 time record, 490, 548
 time reference, lister, 616
 time specified, 768
 time, calibration, 332

- time, mask test run, 819
- time, runt pulse search, 1198
- time, runt trigger, 1534
- time, system, 1460
- time, transition search, 1203
- time, transition trigger, 1545
- time/div, how autoscale affects, 265
- Timebase commands, 1467
- timebase vernier, 1475
- Timebase:MODE, 84
- time-ordered label list, 439
- timeout, SPI clock, 1106
- timezone, 1463
- timing measurement, 671
- title channels, 354
- title, mask test, 827
- token PID, USB search, 1268
- token PID, USB trigger, 1157
- tolerance for determining valid SENT sync pulses, 1092
- tolerance, automask, 803, 804
- top of waveform value measured, 764
- Total Harmonic Distortion, FFT analysis measurement, 692
- total harmonics distortion (THD), 878
- total waveforms in mask test, 808
- touchscreen on/off, 1464
- trace memory, 278
- track measurements, 730
- transfer instrument state, 242, 1457
- transforms, math, 561
- transient response, 791
- transient response analysis, 936, 937, 940
- TRANSition SEARCh commands, 1199
- transition search qualifier, 1200
- transition search slope, 1201
- transition search time, 1203
- transition trigger qualifier, 1542
- transition trigger slope, 1543
- transition trigger source, 1202, 1544
- transition trigger time, 1545
- transparent screen background, remote command logging, 1455
- transparent, display, 449
- tree traversal rules, 1714
- trend measurement, 572
- TRG (Trigger), 252, 254, 256
- trigger armed status, checking for, 1677
- trigger burst, UART, 1130
- trigger channel source, 1519
- TRIGger commands, 1479
- TRIGger commands, general, 1481
- trigger data, UART, 1131
- TRIGger DELay commands, 1497
- trigger duration, 1526, 1527
- TRIGger EDGE commands, 1505
- trigger edge coupling, 1506
- trigger edge slope, 1509
- trigger edge slope, Edge Then Edge trigger, 1503
- trigger edge source, Edge Then Edge trigger, 1504
- trigger event bit, 280
- Trigger Event Register, reading a Condition register bit, 1397
- Trigger Event Register, reading and clearing an Event register bit, 1401
- Trigger Event Register, reading and clearing the Event register, 1406
- Trigger Event Register, reading the Condition register, 1402
- Trigger Event Register, setting and reading a Negative Transition filter bit, 1399
- Trigger Event Register, setting and reading a Positive Transition filter bit, 1400
- Trigger Event Register, setting and reading an Enable register bit, 1398
- Trigger Event Register, setting and reading the Enable register, 1403
- Trigger Event Register, setting and reading the Negative Transition filter, 1404
- Trigger Event Register, setting and reading the Positive Transition filter, 1405
- TRIGger GLITCh commands, 1511
- trigger holdoff, 1485
- trigger hysteresis disable/enable, 1489
- trigger idle, UART, 1132
- TRIGger IIC commands, 1031
- trigger level auto set up, 1491
- trigger level constants, 356
- trigger level voltage, 1507
- trigger level, high, 1492
- trigger level, low, 1493
- TRIGger LIN commands, 1041
- TRIGger M1553 commands, 1059
- trigger occurred, 254
- TRIGger OR commands, 1520
- TRIGger PATtern commands, 1522
- trigger pattern qualifier, 1528
- trigger qualifier, UART, 1133
- trigger reset conditions, 248, 1446
- TRIGger RUNT commands, 1530
- TRIGger SHOLd commands, 1535
- trigger SPI clock slope, 1105
- TRIGger SPI commands, 1102
- trigger status bit, 280
- TRIGger status commands, 1394
- trigger sweep mode, 1479
- TRIGger TV commands, 1541
- trigger type, ARINC 429, 1002, 1206
- trigger type, CAN, 1028
- trigger type, SPI, 1117
- trigger type, UART, 1134
- TRIGger UART commands, 1119
- TRIGger USB commands, 1136
- TRIGger ZONE commands, 1546
- trigger, ARINC 429 source, 995
- trigger, CAN, 1015
- trigger, CAN FD sample point, 1008
- trigger, CAN pattern data length, 1021
- trigger, CAN pattern ID mode, 1024
- trigger, CAN sample point, 1009
- trigger, CAN signal baudrate, 1010
- trigger, CAN signal definition, 1011
- trigger, CAN source, 1014
- trigger, CAN XL sample point, 1030
- trigger, duration greater than, 1526
- trigger, duration less than, 1527
- trigger, duration range, 1529
- trigger, edge coupling, 1506
- trigger, edge level, 1507
- trigger, edge reject, 1508
- trigger, edge slope, 1509
- trigger, edge source, 1510
- trigger, force a, 1483
- trigger, glitch greater than, 1513
- trigger, glitch less than, 1514
- trigger, glitch level, 1515
- trigger, glitch polarity, 1516
- trigger, glitch qualifier, 1517
- trigger, glitch range, 1518
- trigger, glitch source, 1519
- trigger, high frequency reject filter, 1484
- trigger, holdoff, 1485
- trigger, IIC clock source, 1033
- trigger, IIC data source, 1034
- trigger, IIC pattern address, 1035
- trigger, IIC pattern data, 1036
- trigger, IIC pattern data 2, 1037
- trigger, IIC qualifier, 1038
- trigger, IIC signal baudrate, 1046
- trigger, IIC type, 1039
- trigger, LIN, 1050
- trigger, LIN pattern data, 1052
- trigger, LIN pattern data length, 1054
- trigger, LIN pattern format, 1055
- trigger, LIN sample point, 1045
- trigger, LIN source, 1047
- trigger, mode, 1494
- trigger, noise reject filter, 1495
- trigger, SPI clock slope, 1105
- trigger, SPI clock source, 1109
- trigger, SPI clock timeout, 1106
- trigger, SPI frame source, 1110
- trigger, SPI framing, 1108
- trigger, SPI pattern MISO width, 1114
- trigger, SPI pattern MOSI width, 1116
- trigger, sweep, 1496
- trigger, UART base, 1129
- trigger, UART baudrate, 1122
- trigger, UART bit order, 1123
- trigger, UART parity, 1125
- trigger, UART polarity, 1126
- trigger, UART Rx source, 1127
- trigger, UART Tx source, 1128
- trigger, UART width, 1135
- trigger, USB, 1144
- truncation rules, 1710
- TST (Self Test), 257
- TTL threshold voltage for digital channels, 404
- turn off channel, 270
- turn off channel labels, 438
- turn off digital pod, 270
- turn off math function, 270
- turn off time, 782, 890, 891

turn on channel labels, 438
 turn on time, 783, 890, 891
 turn on/turn off analysis thresholds, 891
 turn on/turn off time analysis, 887, 888, 889, 890
 turning channel display on and off, 351
 turning off/on function calculation, 533
 TV trigger commands, 1541
 Tx data, UART, 1581
 Tx source, 1128
 type of power being converted, efficiency measurement, 865
 type, CAN protocol decode, 1029

U

UART base, 1129
 UART baud rate, 1122
 UART bit order, 1123
 UART parity, 1125
 UART polarity, 1126
 UART Rx source, 1127
 UART SEARCh commands, 1249
 UART serial search, data, 1250
 UART serial search, data qualifier, 1253
 UART serial search, mode, 1251
 UART trigger burst, 1130
 UART trigger commands, 1119
 UART trigger data, 1131
 UART trigger idle, 1132
 UART trigger qualifier, 1133
 UART trigger type, 1134
 UART Tx data, 1581
 UART Tx source, 1128
 UART width, 1135
 UART/RS-232 triggering, 984
 units (vertical) for FFT, 489, 547
 units per division, 364, 375, 376, 463, 1471
 units per division (vertical) for FFT function, 485
 units per division (vertical) for function, 375, 567
 units, automask, 802
 units, X cursor, 643, 644
 units, Y cursor, 650, 651
 unsigned data, 1558
 unsigned mode, 1583
 upper threshold, 718
 uppercase characters in commands, 1709
 URQ (User Request) status bit, 238, 240
 USB (Device) interface, 71
 USB 2.0 triggering, 984
 USB PD search, Header Content control message type, 1276
 USB PD search, Header Content data message type, 1278
 USB PD search, Header Content extended message type, 1279
 USB PD search, Header Content header type, 1274
 USB PD search, Header Content qualifier, 1281
 USB PD search, Header Content value, 1282
 USB PD search, mode, 1273
 USB PD trigger mode, 1163
 USB PD trigger, control message type, 1166
 USB PD trigger, data message type, 1168, 1169
 USB PD trigger, header content trigger qualifier, 1172
 USB PD trigger, header type, 1164
 USB PD trigger, user-defined header value, 1171
 USB PD triggering, 984
 USB PD waveform source, 1162
 USB SEARCh commands, 1254
 USB storage device, recalling files from, 943
 USB storage device, saving files to, 956
 USB trigger commands, 1136
 USBPd SEARCh commands, 1272
 USBPd serial bus commands, 1161
 user defined channel labels, 354
 user event conditions occurred, 254
 User Event Register, reading a Condition register bit, 1410
 User Event Register, reading and clearing an Event register bit, 1414
 User Event Register, reading and clearing the Event register, 1419
 User Event Register, reading the Condition register, 1415
 User Event Register, setting and reading a Negative Transition filter bit, 1412
 User Event Register, setting and reading a Positive Transition filter bit, 1413
 User Event Register, setting and reading an Enable register bit, 1411
 User Event Register, setting and reading the Enable register, 1416
 User Event Register, setting and reading the Negative Transition filter, 1417
 User Event Register, setting and reading the Positive Transition filter, 1418
 user interface User memory depth setting, 294
 user interface User sample rate setting, 295
 User memory depth setting, user interface, 294
 User sample rate setting, user interface, 295
 USER status commands, 1407
 User's Guide, 6
 user-defined baud rate, ARINC 429, 992
 user-defined Real Power in Class D harmonics analysis, 874
 user-defined threshold voltage for digital channels, 404
 USR (User Event bit), 252, 254

V

valid command strings, 1709
 valid pattern time, 1526, 1527
 value, 754
 value measured at base of waveform, 758
 value measured at specified time, 768
 value measured at top of waveform, 764
 value ranges, 228
 values required to fill time buckets, 288
 VBA, 80, 1718
 Vce(sat) power measurement, 792
 Vce(sat) value for conduction calculation, 934
 vectors, display, 450
 vectors, turning on or off, 409
 vernier, channel, 377
 vernier, horizontal, 1475
 vertical adjustment, fine (vernier), 377
 vertical amplitude measurement, 756
 vertical axis defined by RANGE, 481, 565
 vertical axis range for channels, 374
 vertical histogram, 580
 vertical offset for channels, 355
 vertical peak-to-peak measured on waveform, 761
 vertical scale, 375, 485, 567
 vertical scaling, 1570
 vertical units for FFT, 489, 547
 vertical value at center screen, 480, 484, 560, 566
 vertical value maximum measured on waveform, 759
 vertical value measurements to calculate overshoot, 715
 vertical value minimum measured on waveform, 760
 view, 281, 1584
 VISA COM example in C#, 1727
 VISA COM example in Python, 1745
 VISA COM example in Visual Basic, 1718
 VISA COM example in Visual Basic .NET, 1736
 VISA example in C, 1752
 VISA example in C#, 1761
 VISA example in Python, 1782
 VISA example in Visual Basic .NET, 1772
 VISA examples, 1718, 1752
 VISA.NET example in C#, 1789
 VISA.NET example in Python, 1801
 VISA.NET example in Visual Basic .NET, 1795
 VISA.NET examples, 1789
 Visual Basic .NET, VISA COM example, 1736
 Visual Basic .NET, VISA example, 1772
 Visual Basic .NET, VISA.NET example, 1795
 Visual Basic 6.0, 81
 Visual Basic for Applications, 80, 1718
 Visual Basic, VISA COM example, 1718
 visualizations, math, 563

voltage (waveform generator), frequency response analysis, 513
 voltage at center screen, 362
 voltage difference between data points, 1588
 voltage in, frequency response analysis, 508, 509
 voltage level for active trigger, 1507
 voltage offset value for channels, 355
 voltage probe, 364, 376, 463
 voltage profile, frequency response analysis, 515
 voltage ranges for channels, 374
 voltage ranges for external trigger, 462
 voltage source, 927
 voltage, maximum expected input, 920, 921, 922

W

WAI (Wait To Continue), 258
 wait, 258
 wait for operation complete, 243
 warranty, 2
 waveform base value measured, 758
 WAVEform command, 79
 WAVEform commands, 1553
 waveform data, 1555
 waveform data format, 976
 waveform data length, 977
 waveform data length, maximum, 978
 waveform data, save, 975
 waveform generator, 1594
 waveform generator amplitude, 513, 1632
 waveform generator function, 1607
 waveform generator high-level voltage, 1633
 waveform generator load impedance, Control Loop Response (Bode) power analysis, 860
 waveform generator load impedance, Power Supply Rejection Ratio (PSRR) power analysis, 904
 waveform generator low-level voltage, 1634
 waveform generator offset, 1635
 waveform generator output control, 1624
 waveform generator output load impedance, 512, 1627
 waveform generator output mode, 1628
 waveform generator output polarity, 1629
 waveform generator period, 1630
 waveform generator pulse width, 1613
 waveform generator ramp symmetry, 1614
 waveform generator reset defaults, 1631
 waveform generator square wave duty cycle, 1615
 waveform introduction, 1555
 waveform maximum vertical value measured, 759
 waveform minimum vertical value measured, 760
 WAVEform parameters, 84
 waveform peak-to-peak vertical value measured, 761
 waveform period, 718
 waveform persistence, 409
 waveform RMS value measured, 763
 waveform save option for segments, 979
 waveform source, 1577
 waveform source subsample, 1581
 waveform standard deviation value measured, 729
 waveform vertical amplitude, 756
 waveform, byte order, 1561
 waveform, count, 1562
 waveform, data, 1563
 waveform, format, 1565
 waveform, points, 1566, 1568
 waveform, preamble, 1570
 waveform, type, 1582
 waveform, unsigned, 1583
 waveform, view, 1584
 waveform, X increment, 1585
 waveform, X origin, 1586
 waveform, X reference, 1587
 waveform, Y increment, 1588
 waveform, Y origin, 1589
 waveform, Y reference, 1590
 WAVEform:FORMat, 84
 waveforms, mask test run, 820
 waven output amplitude(s), Control Loop Response (Bode) power analysis, 861
 waven output amplitude(s), Power Supply Rejection Ratio (PSRR) power analysis, 905
 website, examples on, 1717
 WGEN commands, 1591
 WGEN trigger source, 1510
 what's new, 41
 width, 1135, 1518
 -Width to -Width jitter measurement, 679
 window, 1476, 1477, 1478
 window time, 1470
 window time base mode, 1472
 window, measurement, 765
 windows, 490, 548
 windows as filters to Fast Fourier Transforms, 490, 548
 windows for Fast Fourier Transform functions, 490, 548
 WMEMory commands, 1637
 word format, 1565
 word format for data transfer, 1558
 word format, ARINC 429, 993
 word width, SPI decode, 1118
 write mode, remote command logging, 1450, 1456
 write text to display, 1427
 WriteIEEEBlock method, 81, 87
 WriteList method, 81
 WriteNumber method, 81

WriteString method, 81

X

X axis markers, 633
 X cursor units, 643, 644
 X delta, 634, 642
 X delta, mask scaling, 823
 X1 and X2 cursor value difference, 634, 642
 X1 cursor, 633, 637, 638
 X1, mask scaling, 822
 X2 cursor, 633, 640, 641
 X-axis functions, 1468
 X-increment, 1585
 X-of-max measurement, 766
 X-of-min measurement, 767
 X-origin, 1586
 X-reference, 1587
 XY color scheme, 451
 X-Y mode, 1468, 1472

Y

Y axis markers, 633
 Y cursor units, 650, 651
 Y offset, reference waveform, 1644
 Y range, reference waveform, 1645
 Y scale, reference waveform, 1646
 Y1 and Y2 cursor value difference, 649
 Y1 cursor, 633, 638, 646, 649
 Y1, mask scaling, 824
 Y2 cursor, 633, 641, 648, 649
 Y2, mask scaling, 825
 Y-axis value, 1589
 Y-increment, 1588
 Y-origin, 1589, 1590
 Y-reference, 1590

Z

zero values in waveform data, 1563
 zone combine logic, 1547
 zone mode, 1548
 zone placement, 1549
 zone source, 1550
 zone state, 1551
 ZONE trigger commands, 1546
 zone validity, 1552
 Zoom In/Out setting for N2820A channel, 372
 zoomed time base, 1472
 zoomed time base measurement window, 765
 zoomed time base mode, how autoscale affects, 265
 zoomed window horizontal scale, 1478

