Agilent 6000 Series Oscilloscopes

Programmer's Reference
Notices


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 Agilent Technologies, Inc. as governed by United States and international copyright laws.

Manual Part Number
Version 04.10.0000
Edition
April 26, 2007
Available in electronic format only
Agilent 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, Agilent 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. Agilent 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 Agilent 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 Licenses
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.

Restricted Rights Legend
If software is for use in the performance of a U.S. Government prime contract or sub-contract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015(b)(2) (November 1995), 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.

Trademarks
Microsoft®, MS-DOS®, Windows®, Windows 2000®, and Windows XP® are U.S. registered trademarks of Microsoft Corporation.
Adobe®, Acrobat®, and the Acrobat Logo® are trademarks of Adobe Systems Incorporated.

Technology Licenses

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 programmer's reference gives detailed information on all the commands available for controlling these oscilloscope models:

Table 1  6000 Series Oscilloscope Models

<table>
<thead>
<tr>
<th>Channels</th>
<th>Input Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 GHz</td>
</tr>
<tr>
<td>4 analog + 16 digital</td>
<td>MSO6104A/L</td>
</tr>
<tr>
<td>(mixed-signal)</td>
<td></td>
</tr>
<tr>
<td>2 analog + 16 digital</td>
<td>MSO6102A</td>
</tr>
<tr>
<td>(mixed-signal)</td>
<td></td>
</tr>
<tr>
<td>4 analog</td>
<td>DSO6104A/L</td>
</tr>
<tr>
<td>2 analog</td>
<td>DSO6102A</td>
</tr>
</tbody>
</table>

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.

Command arguments and syntax are described for each command. Some command descriptions have example code.

- "What's New" on page 17
- "Commands Quick Reference" on page 31
- "Commands by Subsystem" on page 73
- "Commands A-Z" on page 469
- "Obsolete and Discontinued Commands" on page 491
- "Error Messages" on page 533
- "Status Reporting" on page 541
- "More About Oscilloscope Commands" on page 563
- "Programming Examples" on page 585

See the Agilent 6000 Series Oscilloscopes Programmer's Quick Start Guide for information on installing the IO libraries, connecting the oscilloscope to the controller PC, and getting started with oscilloscope programming.

See your oscilloscope's User's Guide for more information on front-panel operation.
### Mixed-Signal Oscilloscope Channel Differences

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

### Example Programs

The example programs are designed to work with multiple 6000 Series oscilloscopes. Therefore, the commands may not match the example code exactly, but the example code should run because of the designed-in backward compatibility with earlier commands.
Contents

In This Book 3

1 What’s New

What’s New in Version 4.10 18
What’s New in Version 4.00 20
What’s New in Version 3.50 22
What’s New in Version 3.00 24
Command Differences From 54620/54640 Series Oscilloscopes 26

2 Commands Quick Reference

Command Summary 32
Syntax Elements 70
Number Format 70
<NL> (Line Terminator) 70
[ ] (Optional Syntax Terms) 70
{ } (Braces) 70
 ::= (Defined As) 70
<> (Angle Brackets) 71
... (Ellipsis) 71
n,...,p (Value Ranges) 71
d (Digits) 71
Quoted ASCII String 71
Definite-Length Block Response Data 71

3 Commands by Subsystem

Common (*) Commands 75
*CLS (Clear Status) 79
*ESE (Standard Event Status Enable) 80
*ESR (Standard Event Status Register) 82
*IDN (Identification Number) 84
*LRN (Learn Device Setup) 85
*OPC (Operation Complete) 86
*OPT (Option Identification) 87
*RCL (Recall) 88
*RST (Reset)  89
*SAV (Save)  92
*SRE (Service Request Enable)  93
*STB (Read Status Byte)  95
*TRG (Trigger)  97
*TST (Self Test)  98
*WAI (Wait To Continue)  99

Root (: ) Commands  100

:ACTivity  103
:AER (Arm Event Register)  104
:AUToscale  105
:AUToscale:AMODE  107
:AUToscale:CHANnels  108
:BLANk  109
:CDISplay  110
:DIGitize  111
:HWEnable (Hardware Event Enable Register)  113
:HWERegister:CONDition (Hardware Event Condition Register)  115
:HWERegister[:EVENT] (Hardware Event Event Register)  117
:MERGe  119
:OPEE (Operation Status Enable Register)  120
:OPERegister:CONDition (Operation Status Condition Register)  122
:OPERegister[:EVENT] (Operation Status Event Register)  124
:OVLenable (Overload Event Enable Register)  126
:OVLRegister (Overload Event Register)  128
:PRINt  130
:RUN  131
:SERial  132
:SINGle  133
:STATus  134
:STOP  135
:TER (Trigger Event Register)  136
:VIEW  137

:ACQuire Commands  138
:ACQuire:AAALias  140
:ACQuire:COMPLETE  141
:ACQuire:COUNt  142
:ACQuire:DAALias  143
:ACQuire:MODE  144
:ACQuire:POINts  145
:ACQuire:RSIGnal  146
:MARKer:X2Position 237
:MARKer:X2Y2source 238
:MARKer:XDELta 239
:MARKer:Y1Position 240
:MARKer:Y2Position 241
:MARKer:YDELta 242

:MEASure Commands 243
:MEASure:CLEar 250
:MEASure:COUNter 251
:MEASure:DEFine 252
:MEASure:DELay 255
:MEASure:DUTYcycle 257
:MEASure:FALLtime 258
:MEASure:FREQuency 259
:MEASure:NWIDth 260
:MEASure:OVERshoot 261
:MEASure:PERiod 263
:MEASure:PHASE 264
:MEASure:PREShoot 265
:MEASure:PWIDth 266
:MEASure:RIStime 267
:MEASure:SDEViation 268
:MEASure:SHOW 269
:MEASure:SOURce 270
:MEASure:TEDGe 272
:MEASure:TVALue 274
:MEASure:VAMPplitude 276
:MEASure:VAVerage 277
:MEASure:VBASe 278
:MEASure:VMAX 279
:MEASure:VMIN 280
:MEASure:VPP 281
:MEASure:VRMS 282
:MEASure:VTIME 283
:MEASure:VTOP 284
:MEASure:XMAX 285
:MEASure:XMIN 286

:POD Commands 287
:POD<n>:DISPlay 288
:POD<n>:SIZE 289
:POD<n>:THReshold 290
:SBUS Commands 292
:SBUS:BUSDoctor:ADDRess 294
:SBUS:BUSDoctor:BAUDrate 295
:SBUS:BUSDoctor:CHANnel 296
:SBUS:BUSDoctor:MODE 297
:SBUS:CAN:COUNT:ERRor 298
:SBUS:CAN:COUNT:OVERload 299
:SBUS:CAN:COUNT:RESet 300
:SBUS:CAN:COUNT:TOTal 301
:SBUS:CAN:COUNT:UTILization 302
:SBUS:DISPlay 303
:SBUS:FLEXray:COUNT:NULL 304
:SBUS:FLEXray:COUNT:RESet 305
:SBUS:FLEXray:COUNT:SYNC 306
:SBUS:FLEXray:COUNT:TOTal 307
:SBUS:IIC:ASIZe 308
:SBUS:LIN:PARity 309
:SBUS:MODE 310
:SBUS:SPI:WIDTH 311

:SYSTem Commands 312
:SYSTem:DATE 313
:SYSTem:DSP 314
:SYSTem:ERRor 315
:SYSTem:LOCK 316
:SYSTem:SETup 317
:SYSTem:TIME 319

:TIMebase Commands 320
:TIMebase:MODE 322
:TIMebase:POSition 323
:TIMebase:RANGe 324
:TIMebase:REFClock 325
:TIMebase:REference 326
:TIMebase:SCALE 327
:TIMebase:VERNier 328
:TIMebase:WINDow:POSition 329
:TIMebase:WINDow:RANGe 330
:TIMebase:WINDow:SCALE 331

:TRIGger Commands 332
General :TRIGger Commands 335
:TRIGger:HFReject 336
:TRIGger:HOLDoff 337
:TRIGger:MODE 338
:TRIGger:NREJect 339
:TRIGger:PATTERn 340
:TRIGger:SWEep 342
:TRIGger:CAN Commands 343
:TRIGger:CAN:PATTERn:DATA 345
:TRIGger:CAN:PATTERn:DATA:LENGTH 346
:TRIGger:CAN:PATTERn:ID 347
:TRIGger:CAN:PATTERn:ID:MODE 348
:TRIGger:CAN:SAMPlepoint 349
:TRIGger:CAN:SIGNal:BAUDrate 350
:TRIGger:CAN:SOURce 351
:TRIGger:CAN:TRIGger 352
:TRIGger:DURation Commands 354
:TRIGger:DURation:GREaterthan 355
:TRIGger:DURation:LESSthan 356
:TRIGger:DURation:PATTERn 357
:TRIGger:DURation:QUALifier 358
:TRIGger:DURation:RANGE 359
:TRIGger:EBURst Commands 360
:TRIGger:EBURst:COUNT 361
:TRIGger:EBURst:IDLE 362
:TRIGger:EBURst:SLOPe 363
:TRIGger[:EDGE] Commands 364
:TRIGger[:EDGE]:COUPling 365
:TRIGger[:EDGE]:LEVel 366
:TRIGger[:EDGE]:REJect 367
:TRIGger[:EDGE]:SLOPe 368
:TRIGger[:EDGE]:SOURce 369
:TRIGger:FLEXray Commands 370
:TRIGger:FLEXray:ERRor:TYPE 371
:TRIGger:FLEXray:FRAME:CCBase 373
:TRIGger:FLEXray:FRAME:CCRepetition 374
:TRIGger:FLEXray:FRAME:ID 375
:TRIGger:FLEXray:FRAME:TYPE 376
:TRIGger:FLEXray:TIME:CBASE 377
:TRIGger:FLEXray:TIME:CREPetition 378
:TRIGger:FLEXray:TIME:SEGment 379
:TRIGger:FLEXray:TIME:SLOT 380
:TRIGger:FLEXray:TRIGger 381
:TRIGger:GLITch Commands 382
:TRIGger:TV:MODE 427
:TRIGger:TV:POLarity 428
:TRIGger:TV:SOURce 429
:TRIGger:TV:STANdard 430
:TRIGger:USB Commands 431
:TRIGger:USB:SOURce:DMINus 432
:TRIGger:USB:SOURce:DPLUS 433
:TRIGger:USB:SPEed 434
:TRIGger:USB:TRIGger 435

:WAVeform Commands 436
:WAVeform:BYTorder 444
:WAVeform:COUNt 445
:WAVeform:DATA 446
:WAVeform:FORMat 448
:WAVeform:POINts 449
:WAVeform:POINts:MODE 451
:WAVeform:PREamble 453
:WAVeform:SOURce 456
:WAVeform:TYPE 460
:WAVeform:UNSIGNED 461
:WAVeform:VIEW 462
:WAVeform:XINCrement 463
:WAVeform:XORigin 464
:WAVeform:XREFERence 465
:WAVeform:YINCrement 466
:WAVeform:YORigin 467
:WAVeform:YREFerence 468

4 Commands A-Z

5 Obsolete and Discontinued Commands

:CHANnel:ACTivity 496
:CHANnel:LABel 497
:CHANnel:THReshold 498
:CHANnel2:SKEW 499
:CHANnel<n>:INPut 500
:CHANnel<n>:PMODE 501
:DISPlay:CONNnect 502
:DISPlay:ORDer 503
:ERASE 504
:EXTernal:INPut 505
6 Error Messages

7 Status Reporting

Status Reporting Data Structures  543
Status Byte Register (STB)  546
Service Request Enable Register (SRE)  548
Trigger Event Register (TER)  549
Output Queue  550
Message Queue  551
(Standard) Event Status Register (ESR)  552
(Standard) Event Status Enable Register (ESE)  553
Error Queue  554
Operation Status Event Register (:OPERegister[:EVENT]) 555
Operation Status Condition Register (:OPERegister:CONDition) 556
Arm Event Register (AER) 557
Hardware Event Event Register (:HWERegister[:EVENT]) 558
Hardware Event Condition Register (:HWERegister:CONDition) 559
Clearing Registers and Queues 560
Status Reporting Decision Chart 561

8 More About Oscilloscope Commands
Command Classifications 564
  Core Commands 564
  Non-Core Commands 564
  Obsolete Commands 564
Valid Command/Query Strings 565
  Program Message Syntax 565
  Command Tree 569
  Duplicate Mnemonics 579
  Tree Traversal Rules and Multiple Commands 579
Query Return Values 582
All Oscilloscope Commands Are Sequential 583

9 Programming Examples
SICL Example in C 586
VISA Example in C 595
VISA Example in Visual Basic 604
VISA COM Example in Visual Basic 614

Index
1
What's New

What's New in Version 4.10  18
What's New in Version 4.00  20
What’s New in Version 3.50  22
What's New in Version 3.00  24
Command Differences From 54620/54640 Series Oscilloscopes  26
What's New

What's New in Version 4.10

New features in version 4.10 of the 6000 Series oscilloscope software are:

- The ability to trigger on and decode FlexRay serial bus data using a
  Deconsys BusDoctor 2 protocol analyzer with a four-channel
  mixed-signal oscilloscope that includes the Option FRS license.
- The square root waveform math function.
- Several new hardcopy printer drivers.

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

New Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:BUSDoctor:ADDRESS</td>
<td>Sets/queries the four fields in the BusDoctor LAN IP Address.</td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:BAUDrate</td>
<td>Sets/queries the baud rate for the BusDoctor from 2.5 Mb/s to 10 Mb/s.</td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:CHANnel</td>
<td>Sets/queries the FlexRay channel that the BusDoctor analyzes/preprocesses.</td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:MODE</td>
<td>Sets/queries the operating mode of the BusDoctor.</td>
</tr>
<tr>
<td>:SBUS:FLEXray:COUNt:RESet</td>
<td>Resets the FlexRay frame counters.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:ERRor:TYPE</td>
<td>Sets/queries the FlexRay error type to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:CCBase</td>
<td>Sets/queries the base of the FlexRay cycle count (in the frame header) to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:CCRepetition</td>
<td>Sets/queries the repetition number of the FlexRay cycle count (in the frame header) to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:ID</td>
<td>Sets/queries the FlexRay frame ID to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:TYPE</td>
<td>Sets/queries the FlexRay frame type to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:CBASe</td>
<td>Sets/queries the base of the FlexRay cycle to trigger on.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:CREPetition</td>
<td>Sets/queries the repetition number of the FlexRay cycle to trigger on.</td>
</tr>
</tbody>
</table>
### Changed Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:FLEXray:TIME:SEGment (see page 379)</td>
<td>Sets/queries the FlexRay segment type.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:SLOT (see page 380)</td>
<td>Sets/queries the FlexRay slot type and ID.</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TRIGger (see page 381)</td>
<td>Sets/queries the FlexRay trigger mode.</td>
</tr>
<tr>
<td>:FUNCTION:OPERation (see page 217)</td>
<td>You can now select the SQRT (square root) waveform math function.</td>
</tr>
<tr>
<td>:SBUS:MODE (see page 310)</td>
<td>You can now select the FLEXray serial bus decode mode.</td>
</tr>
<tr>
<td>:TRIGger:MODE (see page 338)</td>
<td>You can now select the FLEXray trigger mode.</td>
</tr>
<tr>
<td>:HARDcopy:PDRiver (see page 231)</td>
<td>You can now select the new DJPR0kx50, DJ55xx, PS470, and LJFastraster printer drivers.</td>
</tr>
</tbody>
</table>
What's New in Version 4.00

New features in version 4.00 of the 6000 Series oscilloscope software are:

- The ability to :AUToscale selected channels only and specify the acquisition type and mode that is set after an :AUToscale.
- The :BUS command subsystem for controlling up to two buses made up of digital channels.
- Additional :CALibrate commands for starting the user calibration procedure, displaying the status of the last user calibration, and displaying the temperature change since the last user calibration.

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

### New Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:AUToscale:AMODE (see page 107)</td>
<td>Specifies whether to keep the current acquisition type and mode after subsequent autoscales.</td>
</tr>
<tr>
<td>:AUToscale:CHANnels (see page 108)</td>
<td>Specifies whether to autoscale the currently displayed channels or all channels.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:BIT&lt;m&gt; (see page 152)</td>
<td>Includes or excludes the selected bit in a bus definition.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:BITS (see page 153)</td>
<td>Includes or excludes a list of bits in a bus definition.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:CLEar (see page 155)</td>
<td>Excludes all digital channels from a bus definition.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:DISPLAY (see page 156)</td>
<td>Displays or hides the bus on the oscilloscope display.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:LABEL (see page 157)</td>
<td>Assigns a label string to a bus.</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:MASK (see page 158)</td>
<td>Includes or excludes bits in a bus definition according to a mask.</td>
</tr>
<tr>
<td>:CALibrate:START (see page 162)</td>
<td>Starts the user calibration procedure.</td>
</tr>
<tr>
<td>:CALibrate:STATUS? (see page 163)</td>
<td>Displays the summary results of the last user calibration procedure.</td>
</tr>
<tr>
<td>:CALibrate:TEMPerature? (see page 165)</td>
<td>Displays the change in temperature since the last user calibration procedure.</td>
</tr>
</tbody>
</table>
### Changed Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:AUToscale (see page 105)</td>
<td>You can now specify which channels to autoscale.</td>
</tr>
<tr>
<td>:BLANK (see page 109)</td>
<td>Now, you can also use this command with digital channel buses.</td>
</tr>
<tr>
<td>:DIGitize (see page 111)</td>
<td>Now, you can also use this command with digital channel buses.</td>
</tr>
<tr>
<td>:STATus (see page 134)</td>
<td>Now, you can also use this command with digital channel buses.</td>
</tr>
<tr>
<td>:VIEW (see page 137)</td>
<td>Now, you can also use this command with digital channel buses.</td>
</tr>
<tr>
<td>:WAVEform:SOURce (see page 456)</td>
<td>Now, you can also use this command with digital channel buses.</td>
</tr>
</tbody>
</table>
What's New in Version 3.50

New features in version 3.50 of the 6000 Series oscilloscope software are:

- The CAN and LIN options have been added to the :SBUS:MODE (serial decode mode) command.
- The :SBUS:CAN:COUNt commands have been added to count CAN bus frames, count load utilization, and reset the counters.
- The ALLerrors, OVERload, and ACKerror options have been added to the :TRIGger:CAN:TRIGger command.
- The :SBUS:LIN:PARity command has been added.
- The ID (for Frame Id) option has been added to the :TRIGger:LIN:TRIGger command.
- The :HWERegister:CONDition, :HWERegister[:EVENt], and :HWE commands for the hardware event condition, event, and enable registers have been added.

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

New Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:CAN:COUNt:ERRor? (see page 298)</td>
<td>Returns the CAN bus error frame count.</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNt:OVERload? (see page 298)</td>
<td>Returns the CAN bus overload frame count.</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNt:RESet (see page 298)</td>
<td>Resets the CAN bus counters.</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNt:TOTal? (see page 298)</td>
<td>Returns the CAN bus total frame count.</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNt:UTILization? (see page 298)</td>
<td>Returns a percentage showing CAN bus utilization.</td>
</tr>
<tr>
<td>:SBUS:IIC:ASIZe (see page 308)</td>
<td>Determines whether the Read/Write bit is included as the LSB in the display of the IIC address field of the decode bus.</td>
</tr>
<tr>
<td>:SBUS:LIN:PARity (see page 309)</td>
<td>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.</td>
</tr>
<tr>
<td>:TRIGger:LIN:ID (see page 401)</td>
<td>Defines the LIN identifier searched for in each CAN message when the LIN trigger mode is set to frame ID.</td>
</tr>
</tbody>
</table>
### Changed Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:LIN:SAMPLEpoint (see page 402)</td>
<td>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.</td>
</tr>
<tr>
<td>:TRIGger:LIN:STANDARD (see page 405)</td>
<td>Sets the LIN standard in effect for triggering and decoding to be LIN1.3 or LIN2.0.</td>
</tr>
<tr>
<td>:TRIGger:LIN:SYNCbreak (see page 406)</td>
<td>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.</td>
</tr>
<tr>
<td>:HWEnable (see page 113)</td>
<td>Sets or reads the hardware event enable mask register.</td>
</tr>
<tr>
<td>:HWERegister:CONDITION? (see page 115)</td>
<td>Queries the hardware event condition register.</td>
</tr>
<tr>
<td>:HWERegister[:EVENT]? (see page 117)</td>
<td>Queries the hardware event event register.</td>
</tr>
</tbody>
</table>

### Obsolete Commands

<table>
<thead>
<tr>
<th>Obsolete Command</th>
<th>Current Command Equivalent</th>
<th>Behavior Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:CAN:SIGNal:DEFinition (see page 529)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:SIGNal:DEFinition (see page 530)</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>
What's New in Version 3.00

New features in version 3.00 of the 6000 Series oscilloscope software are:

- The :SBUS command subsystem for controlling serial decode bus display, mode, and other options.
- The EBURst trigger mode and supporting :TRIGger:EBURst commands.
- The :ACQuire:AALiAs and :ACQuire:DAALiAs commands.
- The :MEASure:SDEViation command.
- The :TIMebase:REFClock command.
- Changes to the :TRIGger:IIC commands.
- Changes to the :TRIGger:SEQUence:TRIGger command.
- Changes to the :ACQuire:TYPE and :WAVeform:TYPE commands to add HRESolution type.
- Changes to the :BLANK, :DIGitize, :STATus, :VIEW, and :WAVeform:SOURce commands to include the serial decode bus.
- Changes to the :HARDcopy:FORMat command to support the PNG, ASCiixy, and BINary format types.
- Changes to the :DISPlay:DATA? query and the :PRINt command to support the PNG format.
- Changes to the :WAVeform:POINts command to set from 2000 to 8,000,000 points (in 1-2-5 sequence) when the waveform points mode is MAXimum or RAW.

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

### New Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACQuire:AALiAs? (see page 140)</td>
<td>Returns the current state of the oscilloscope's anti-alias control.</td>
</tr>
<tr>
<td>:ACQuire:DAALiAs (see page 143)</td>
<td>Sets the oscilloscope's disable anti-alias mode.</td>
</tr>
<tr>
<td>:MEASure:SDEViation (see page 268)</td>
<td>Measures the std deviation of a waveform.</td>
</tr>
<tr>
<td>:SBUS:DISPlay (see page 303)</td>
<td>Controls the decoded serial bus display.</td>
</tr>
<tr>
<td>:SBUS:MODE (see page 310)</td>
<td>Determines the decode mode for the serial bus.</td>
</tr>
<tr>
<td>:SBUS:SPI:WIDTH (see page 311)</td>
<td>Determines the number of bits in a word of decoded data for SPI.</td>
</tr>
<tr>
<td>:TIMebase:REFClock (see page 325)</td>
<td>Enables or disables the 10 MHz REF BNC input/output.</td>
</tr>
</tbody>
</table>
## Changed Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:EBURst:COUNt (see page 361)</td>
<td>Sets the Nth edge of burst edge counter resource.</td>
</tr>
<tr>
<td>:TRIGger:EBURst:IDLE (see page 362)</td>
<td>Sets the Nth edge in a burst idle resource.</td>
</tr>
<tr>
<td>:TRIGger:EBURst:SLOPe (see page 361)</td>
<td>Specifies whether the rising edge (POSitive) or falling edge (NEGative) of the Nth edge in a burst will generate a trigger.</td>
</tr>
<tr>
<td>:WAVeform:POINTs:MODE (see page 451)</td>
<td>Sets the waveform points mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACQuire:TYPE (see page 148)</td>
<td>The HRESolution type has been added for smoothing at slower sweep speeds.</td>
</tr>
<tr>
<td>:BLANk (see page 109)</td>
<td>Now, you can also use this command with the serial decode bus.</td>
</tr>
<tr>
<td>:DIGitize (see page 111)</td>
<td>Now, you can also use this command with the serial decode bus.</td>
</tr>
<tr>
<td>:DISPlay:DATA (see page 196)</td>
<td>Now, the PNG format is supported in the query.</td>
</tr>
<tr>
<td>:HARDcopy:FORMat (see page 228)</td>
<td>Now, the PNG, ASCiixy, and BINary formats are also supported.</td>
</tr>
<tr>
<td>:PRINt (see page 130)</td>
<td>Now, the PNG option is supported</td>
</tr>
<tr>
<td>:STATus (see page 134)</td>
<td>Now, you can also use this command with the serial decode bus.</td>
</tr>
<tr>
<td>:TRIGger:IIC:TRIGger[:TYPE] (see page 398)</td>
<td>The ANACknowledge, R7Data2, and W7Data2 types have been added.</td>
</tr>
<tr>
<td>:TRIGger:MODE (see page 338)</td>
<td>The EBURst mode has been added.</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:TRIGger (see page 415)</td>
<td>The EDGE2,COUNT,NREFind (no re-find) option has been added.</td>
</tr>
<tr>
<td>:VIEW (see page 137)</td>
<td>Now, you can now use this command with the serial decode bus.</td>
</tr>
<tr>
<td>:WAVeform:POINTs (see page 449)</td>
<td>Now, you can set from 2000 to 8,000,000 points (in 1-2-5 sequence) when the waveform points mode is MAXimum or RAW.</td>
</tr>
<tr>
<td>:WAVeform:SOURce (see page 456)</td>
<td>Now, you can also use this command with the serial decode bus.</td>
</tr>
<tr>
<td>:WAVeform:TYPE (see page 460)</td>
<td>The HRESolution type has been added for smoothing at slower sweep speeds.</td>
</tr>
</tbody>
</table>
1 What's New

Command Differences From 54620/54640 Series Oscilloscopes

The main differences between the version 1.00 programming command set for the 6000 Series oscilloscopes and the 54620/54640 Series oscilloscopes are related to:

- :HARDcopy and :DISPlay command subsystem changes for USB printers and the high resolution color display.
- New standards supported by the :TRIGger:TV commands.
- Support for 113xA Series probes.
- New "RAW" :WAVeform:POINts option for retrieving raw acquisition record data.
- Discontinuance of the common commands for macros.

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

New Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACQuire:RSIGnal (see page 146)</td>
<td>Selects the 10 MHz reference signal mode.</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:ID? (see page 178)</td>
<td>Returns the type of probe attached to the specified oscilloscope channel.</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:STYPe (see page 180)</td>
<td>Sets the channel probe signal type (STYPe) to differential or single-ended when using the 113xA Series probes, and determines how offset is applied.</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:VERNier (see page 185)</td>
<td>Specifies whether the channel's vernier (fine vertical adjustment) setting is ON (1) or OFF (0).</td>
</tr>
<tr>
<td>:DIGital&lt;n&gt;:SIZE (see page 191)</td>
<td>Specifies the size of digital channels on the display.</td>
</tr>
<tr>
<td>:EXTernal:PROBe:ID (see page 207)</td>
<td>Returns the type of probe attached to the external trigger input.</td>
</tr>
<tr>
<td>:EXTernal:PROBe:STYPe (see page 208)</td>
<td>Sets the external trigger probe signal type (STYPe) to differential or single-ended when using the 113xA Series probes, and determines how offset is applied.</td>
</tr>
<tr>
<td>:HARDcopy:FIlename (see page 227)</td>
<td>Sets the output filename for print formats whose output is a file. Replaces the 5462x/4x :HARDcopy:DEStination (see page 508) command.</td>
</tr>
<tr>
<td>:HARDcopy:PDRiver (see page 231)</td>
<td>Sets the hardcopy printer driver.</td>
</tr>
<tr>
<td>:HARDcopy:IGColors (see page 229)</td>
<td>Specifies whether graticule colors are inverted.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>:HARDcopy:PALette (see page 230)</td>
<td>Sets the hardcopy palette color. Replaces the 5462x/4x :HARDcopy:GRAYscale (see page 510) command.</td>
</tr>
<tr>
<td>:OPERegister:CONDition? (see page 122)</td>
<td>Returns the integer value contained in the &quot;Operation Status Condition Register&quot; on page 122 (a new register in addition to the &quot;Operation Status Event Register&quot; on page 124 whose value is returned by the :OPERegister[:EVENT]? (see page 124) query).</td>
</tr>
<tr>
<td>:POD&lt;n&gt;:SIZE (see page 289)</td>
<td>Specifies the size of digital channels on the display.</td>
</tr>
<tr>
<td>:TIMebase:VERNier (see page 328)</td>
<td>Specifies whether the time base control's vernier (fine horizontal adjustment) setting is ON (1) or OFF (0).</td>
</tr>
</tbody>
</table>

**Changed Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Differences From 5462x/4x Oscilloscopes</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACQuire:COUNt (see page 142)</td>
<td>The count can be set to any value from 1 to 65536 (instead of 16383).</td>
</tr>
<tr>
<td>:DISPlay:DATA (see page 196)</td>
<td>The BMP8bit &lt;format&gt; option has been added to the query. There is a new &lt;palette&gt; option which can be MONochrome, GRAYscale, or COlor in the query, or just MONochrome in the command.</td>
</tr>
<tr>
<td>:DISPlay:SOURce (see page 201)</td>
<td>The number of pixel memory locations is 10 (instead of 3).</td>
</tr>
<tr>
<td>:HARDcopy:FORMat (see page 228)</td>
<td>There is now the BMP8bit format (instead of TIFF) and the PRINter0 or PRINter1 formats (in place of LASerjet, DESKjet, EPSon, or SEIKo). See the new :HARDcopy:PDRiver (see page 231) command for setting the hardcopy printer driver.</td>
</tr>
<tr>
<td>*LRN (see page 85)</td>
<td>The Learn Device Setup query return format matches the IEEE 488.2 specification which says that the query result must contain &quot;:SYST:SET &quot; before the binary block data. (This was not the case in the 5462x/4x oscilloscopes.)</td>
</tr>
<tr>
<td>:MERGe (see page 119)</td>
<td>The number of pixel memory locations is 10 (instead of 3).</td>
</tr>
</tbody>
</table>
## What’s New

<table>
<thead>
<tr>
<th>Command</th>
<th>Differences From 5462x/4x Oscilloscopes</th>
</tr>
</thead>
<tbody>
<tr>
<td>*OPT (see page 87)</td>
<td>The Option Identification query return format now has license information (in addition to the I/O module ID information fields which are now always zero).</td>
</tr>
<tr>
<td>:OVLRegister (see page 128)</td>
<td>The Overload Event Register is now a 16-bit register (instead of 8-bit) and it contains bits that identify when faults occur on the oscilloscope channels (in addition to the bits that identify when overloads occur).</td>
</tr>
<tr>
<td>:PRINt (see page 130)</td>
<td>The options are now: COLor (instead of HIRes), GRAYscale (instead of LORes), PRINter0 (instead of PARallel), BMP8bit (instead of TIFF). (The PCL option is now invalid.)</td>
</tr>
<tr>
<td>*RCL (Recall) (see page 88)</td>
<td>The number of instrument state locations is 10 (instead of 3 for the 54620 Series oscilloscopes or 4 for the 54640 Series oscilloscopes).</td>
</tr>
<tr>
<td>*SAV (Save) (see page 92)</td>
<td>The number of instrument state locations is 10 (instead of 3 for the 54620 Series oscilloscopes or 4 for the 54640 Series oscilloscopes).</td>
</tr>
<tr>
<td>*TRG (Trigger) (see page 97)</td>
<td>The *TRG has the same effect as the :DIGitize command with no parameters (instead of the :RUN command).</td>
</tr>
<tr>
<td>:TRIGger:TV:MODE (see page 427)</td>
<td>The modes have been renamed (however, old forms of the mode names are still accepted).</td>
</tr>
<tr>
<td>:TRIGger:TV:STANdard (see page 430)</td>
<td>The P480L60HZ, P720L60HZ, P1080L24HZ, P1080L25HZ, I1080L50HZ, and I1080L60HZ standards are supported (in addition to GENeric, NTSC, PALM, PAL, and SECam).</td>
</tr>
<tr>
<td>:VIEW (see page 137)</td>
<td>The number of pixel memory locations is 10 (instead of 3).</td>
</tr>
<tr>
<td>:WAVeform:COUNt? (see page 445)</td>
<td>The count can be any value from 1 to 65536 (instead of 16383).</td>
</tr>
<tr>
<td>:WAVeform:POINts (see page 449)</td>
<td>There is a new RAW “number of points” option for retrieving the raw acquisition record data. Also the maximum number of points that can be retrieved from the normal measurement record is 1000 (instead of 2000).</td>
</tr>
<tr>
<td>:WAVeform:PREamble (see page 453)</td>
<td>The xincrement format is 64-bit floating point NR3 (instead of 32-bit), and the yreference format is 32-bit NR1 (instead of 16-bit).</td>
</tr>
</tbody>
</table>
### What's New

#### Obsolete Commands

<table>
<thead>
<tr>
<th>Obsolete Command</th>
<th>Current Command Equivalent</th>
<th>Behavior Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:HARDCopy:DESTination</td>
<td>:HARDCopy:FILename</td>
<td>The options are now: COLor (instead of HRes), GRAYscale (instead of LRes), PRINter0 (instead of PARallel), BMP8bit (instead of TIFF). (The DISK and PCL options are now invalid.)</td>
</tr>
<tr>
<td>:HARDCopy:GRAYscale</td>
<td>:HARDCopy:PALETTE</td>
<td></td>
</tr>
<tr>
<td>:PRINT? (see page 525)</td>
<td>:DISPLAY:DATA? (see page 196)</td>
<td></td>
</tr>
</tbody>
</table>

#### Discontinued Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*DMC</td>
<td>Define Macro.</td>
</tr>
<tr>
<td>*EMC</td>
<td>Enable Macro.</td>
</tr>
<tr>
<td>*GMC</td>
<td>Get Macro Contents.</td>
</tr>
<tr>
<td>*LMC</td>
<td>Learn Macro.</td>
</tr>
<tr>
<td>*PMC</td>
<td>Purge Macro.</td>
</tr>
</tbody>
</table>
1 What's New
2

Commands Quick Reference

Command Summary  32
Syntax Elements  70
## Command Summary

### Table 2  Common (*) Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS (see page 79)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>*ESE &lt;mask&gt; (see page 80)</td>
<td>*ESE? (see page 81)</td>
<td>&lt;mask&gt; := 0 to 255; an integer in NR1 format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit Weight Name Enables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- ------ ---- ----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 128 PON Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 64 URQ User Request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 32 CME Command Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 16 EXE Execution Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 8 DDE Dev. Dependent Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 4 QYE Query Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 2 RQL Request Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1 OPC Operation Complete</td>
</tr>
<tr>
<td>n/a</td>
<td>*ESR? (see page 82)</td>
<td>&lt;status&gt; := 0 to 255; an integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>*IDN? (see page 82)</td>
<td>AGILENT TECHNOLOGIES,&lt;model&gt;,&lt;serial number&gt;,X.XX.XX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;model&gt; := the model number of the instrument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;serial number&gt; := the serial number of the instrument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;X.XX.XX&gt; := the software revision of the instrument</td>
</tr>
<tr>
<td>n/a</td>
<td>*LRN? (see page 85)</td>
<td>&lt;learn_string&gt; := current instrument setup as a block of data in IEEE 488.2 # format</td>
</tr>
<tr>
<td>*OPC (see page 86)</td>
<td>*OPC? (see page 86)</td>
<td>ASCII &quot;1&quot; is placed in the output queue when all pending device operations have completed.</td>
</tr>
</tbody>
</table>

32  Agilent 6000 Series Oscilloscopes Programmer’s Reference
Table 2  Common (*) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| n/a              | *OPT? (see page 87) | <return_value> ::= 0,0,<license info>  
                  |                              | <license info> ::= <All field>, <reserved>, <Factory MSO>, <Upgraded MSO>, <Probe field>, <Memory>, <Low Speed Serial>, <reserved>, <reserved>  
                  |                              | <All field> ::= (0 | All)  
                  |                              | <reserved> ::= 0  
                  |                              | <Factory MSO> ::= (0 | MSO)  
                  |                              | <Upgraded MSO> ::= (0 | MSO)  
                  |                              | <Probe field> ::= 0  
                  |                              | <Memory> ::= (0 | mem2M | mem8M)  
                  |                              | <Low Speed Serial> ::= (0 | LSS)  
                  |                              | <reserved> ::= 0  
                  |                              | <reserved> ::= 0  |
| *RCL <value> (see page 88) | n/a | <value> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) |
| *RST (see page 89) | n/a | See *RST (Reset) (see page 89) |
| *SAV <value> (see page 92) | n/a | <value> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) |
| *SRE <mask> (see page 93) | *SRE? (see page 94) | <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  |
Table 2  Common (*) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>*STB? (see page 95)</td>
<td>&lt;value&gt; ::= 0 to 255; an integer in NR1 format, as shown in the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit Weight Name <em>1</em> Indicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- ------ ---- ---------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7  128 OPER Operation status condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6  64  RQS/ MSS requesting service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5  32  ESB Enabled event status condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4  16  MAV Message available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3  8   ---- (Not used.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2  4   MSG Message displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1  2   USR User event condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  1   TRG A trigger occurred.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>*TRG (see page 97)</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>*TST? (see page 98)</td>
<td>&lt;result&gt; ::= 0 or non-zero value; an integer in NR1 format</td>
</tr>
</tbody>
</table>

Table 3  Root (: ) Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACTivity (see page 103)</td>
<td>:ACTivity? (see page 103)</td>
<td>&lt;return value&gt; ::= &lt;edges&gt;,&lt;levels&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;edges&gt; ::= presence of edges (32-bit integer in NR1 format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;levels&gt; ::= logical highs or lows (32-bit integer in NR1 format)</td>
</tr>
<tr>
<td>n/a</td>
<td>:AER? (see page 104)</td>
<td>(0</td>
</tr>
<tr>
<td>:AUToscale [&lt;source&gt;[,...,&lt;source&gt;] ] (see page 105)</td>
<td>n/a</td>
<td>&lt;source&gt; ::= CHANnel&lt;n&gt; for DSO models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; can be repeated up to 5 times</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:AUToscale:AMODE &lt;value&gt; (see page 107)</td>
<td>:AUToscale:AMODE? (see page 107)</td>
<td>&lt;value&gt; ::= (NORMal</td>
</tr>
</tbody>
</table>
### Commands Quick Reference 2

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:AUToscale:CHANnels &lt;value&gt; (see page 108)</td>
<td>:AUToscale:CHANnels? (see page 108)</td>
<td>&lt;value&gt; ::= (ALL</td>
</tr>
<tr>
<td>:BLANk [&lt;source&gt;] (see page 109)</td>
<td>n/a</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;)</td>
</tr>
<tr>
<td>:CDISplay (see page 110)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>:DIGitize [&lt;source&gt;[,...,&lt;source&gt;]] (see page 111)</td>
<td>n/a</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;)</td>
</tr>
<tr>
<td>:HWEnable &lt;n&gt; (see page 113)</td>
<td>:HWEnable? (see page 113)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:HWERegister:CONDition ? (see page 115)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:HWERegister[:EVENt]? (see page 117)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
<tr>
<td>:MERGe &lt;pixel memory&gt; (see page 119)</td>
<td>n/a</td>
<td>&lt;pixel memory&gt; ::= (PMEMory{0</td>
</tr>
<tr>
<td>:OPEE &lt;n&gt; (see page 120)</td>
<td>:OPEE? (see page 121)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:OPERegister:CONDition ? (see page 122)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:OPERegister[:EVENt]? (see page 124)</td>
<td>&lt;n&gt; ::= 16-bit integer in NR1 format</td>
</tr>
</tbody>
</table>
### Table 3  Root (:) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:OVLenable &lt;mask&gt; (see page 126)</td>
<td>:OVLenable? (see page 127)</td>
<td>&lt;mask&gt; ::= 16-bit integer in NR1 format as shown:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit Weight Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- ------ ----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10  1024 Ext Trigger Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9   512  Channel 4 Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8   256  Channel 3 Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7   128  Channel 2 Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6    64  Channel 1 Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4    16  Ext Trigger OVL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3    8   Channel 4 OVL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2    4   Channel 3 OVL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1    2   Channel 2 OVL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0    1   Channel 1 OVL</td>
</tr>
<tr>
<td>n/a</td>
<td>:OVLRegister? (see page 128)</td>
<td>&lt;value&gt; ::= integer in NR1 format. See OVLenable for &lt;value&gt;</td>
</tr>
<tr>
<td>:PRINT [&lt;options&gt;] (see page 130)</td>
<td>n/a</td>
<td>&lt;options&gt; ::= [&lt;print option&gt;],&lt;print option&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;print option&gt; ::= (COlor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;print option&gt; can be repeated up to 5 times.</td>
</tr>
<tr>
<td>:RUN (see page 131)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:SERial (see page 132)</td>
<td>&lt;return value&gt; ::= unquoted string containing serial number</td>
</tr>
<tr>
<td>:SINGLE (see page 133)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:STATus? &lt;display&gt; (see page 134)</td>
<td>(0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;display&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:STOP (see page 135)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
### Table 3  Root (:) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:TER? (see page 136)</td>
<td>(0</td>
</tr>
<tr>
<td>:VIEW &lt;source&gt;</td>
<td>n/a</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>

### Table 4  :ACQuire Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:ACQuire:AALias? (see page 140)</td>
<td>(1</td>
</tr>
<tr>
<td>:ACQuire:COMPLETE &lt;complete&gt; (see page 141)</td>
<td>:ACQuire:COMPLETE? (see page 141)</td>
<td>&lt;complete&gt; ::= 100; an integer in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:COUNT &lt;count&gt; (see page 142)</td>
<td>:ACQuire:COUNT? (see page 142)</td>
<td>&lt;count&gt; ::= an integer from 1 to 65536 in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:DAALias &lt;mode&gt; (see page 143)</td>
<td>:ACQuire:DAALias? (see page 143)</td>
<td>&lt;mode&gt; ::= (DISable</td>
</tr>
<tr>
<td>:ACQuire:MODE &lt;mode&gt; (see page 144)</td>
<td>:ACQuire:MODE? (see page 144)</td>
<td>&lt;mode&gt; ::= (RTIMe</td>
</tr>
<tr>
<td>n/a</td>
<td>:ACQuire:POINTs? (see page 145)</td>
<td># points ::= an integer in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:RSIGnal &lt;ref_signal_mode&gt; (see page 146)</td>
<td>:ACQuire:RSIGnal? (see page 146)</td>
<td>&lt;ref_signal_mode&gt; ::= (OFF</td>
</tr>
<tr>
<td>n/a</td>
<td>:ACQuire:SRATe? (see page 147)</td>
<td>sample_rate ::= sample rate (samples/s) in NR3 format</td>
</tr>
<tr>
<td>:ACQuire:TYPE &lt;type&gt; (see page 148)</td>
<td>:ACQuire:TYPE? (see page 148)</td>
<td>&lt;type&gt; ::= (NORMal</td>
</tr>
</tbody>
</table>
### Table 5  :BUS<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :BUS<n>::BIT<m> {{0 | OFF} | {1 | ON}} (see page 152) | :BUS<n>::BIT<m>? (see page 152) | (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 153) | :BUS<n>::BITS? (see page 153) | <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 155) | n/a | <n> ::= 1 or 2; an integer in NR1 format  
| :BUS<n>::DISPlay {{0 | OFF} | {1 | ON}} (see page 156) | :BUS<n>::DISPlay? (see page 156) | (0 | 1)  
<n> ::= 1 or 2; an integer in NR1 format  
| :BUS<n>::LABel <string> (see page 157) | :BUS<n>::LABel? (see page 157) | <string> ::= quoted ASCII string up to 16 characters  
<n> ::= 1 or 2; an integer in NR1 format  
| :BUS<n>::MASK <mask> (see page 158) | :BUS<n>::MASK? (see page 158) | <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

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:CALibrate:DATE? (see page 160)</td>
<td>&lt;return value&gt; ::= &lt;day&gt;,&lt;month&gt;,&lt;year&gt;; all in NR1 format</td>
</tr>
<tr>
<td>:CALibrate:LABel</td>
<td>:CALibrate:LABel? (see page 161)</td>
<td>&lt;string&gt; ::= quoted ASCII string up to 32 characters</td>
</tr>
<tr>
<td>:CALibrate:STARt</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:STATUS? (see page 163)</td>
<td>&lt;return value&gt; ::= ALL,&lt;status_code&gt;,&lt;status_string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;status_code&gt; ::= an integer status code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;status_string&gt; ::= an ASCII status string</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:SWITch? (see page 164)</td>
<td>{PROTected</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:TEMPerature? (see page 165)</td>
<td>&lt;return value&gt; ::= degrees C delta since last cal in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:TIME? (see page 166)</td>
<td>&lt;return value&gt; ::= &lt;hours&gt;,&lt;minutes&gt;,&lt;seconds&gt;; all in NR1 format</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CHANnel&lt;n&gt;:BWLimit</td>
<td>:CHANnel&lt;n&gt;:BWLimit? (see page 170)</td>
<td>(0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:COUPling</td>
<td>:CHANnel&lt;n&gt;:COUPling? (see page 171)</td>
<td>&lt;coupling&gt; ::= (AC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:DISPlay</td>
<td>:CHANnel&lt;n&gt;:DISPlay? (see page 172)</td>
<td>(0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:IMPedance</td>
<td>:CHANnel&lt;n&gt;:IMPedance? (see page 173)</td>
<td>&lt;impedance&gt; ::= (ONEMeg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:INVert</td>
<td>:CHANnel&lt;n&gt;:INVert? (see page 174)</td>
<td>(0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>
## Commands Quick Reference

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CHANnel&lt;n&gt;:LABel &lt;string&gt; (see page 175)</td>
<td>:CHANnel&lt;n&gt;:LABel? (see page 175)</td>
<td>&lt;string&gt; ::= any series of 6 or less ASCII characters enclosed in quotation marks&lt;br&gt;&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:OFFSet &lt;offset&gt;[suffix] (see page 176)</td>
<td>:CHANnel&lt;n&gt;:OFFSet? (see page 176)</td>
<td>&lt;offset&gt; ::= Vertical offset value in NR3 format&lt;br&gt;[suffix] ::= (V</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe &lt;attenuation&gt; (see page 177)</td>
<td>:CHANnel&lt;n&gt;:PROBe? (see page 177)</td>
<td>&lt;attenuation&gt; ::= Probe attenuation ratio in NR3 format&lt;br&gt;&lt;n&gt; ::= 1-2 or 1-4r in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:CHANnel&lt;n&gt;:PROBe:ID? (see page 178)</td>
<td>&lt;probe id&gt; ::= unquoted ASCII string up to 11 characters&lt;br&gt;&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:SKEW &lt;skew_value&gt; (see page 179)</td>
<td>:CHANnel&lt;n&gt;:PROBe:SKEW? (see page 179)</td>
<td>&lt;skew_value&gt; ::= -100 ns to +100 ns in NR3 format&lt;br&gt;&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:STYPe &lt;signal type&gt; (see page 180)</td>
<td>:CHANnel&lt;n&gt;:PROBe:STYPe? (see page 180)</td>
<td>&lt;signal type&gt; ::= {DIFFerential</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROTection (see page 181)</td>
<td>:CHANnel&lt;n&gt;:PROTection? (see page 181)</td>
<td>{NORM</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:RANGe &lt;range&gt;[suffix] (see page 182)</td>
<td>:CHANnel&lt;n&gt;:RANGe? (see page 182)</td>
<td>&lt;range&gt; ::= Vertical full-scale range value in NR3 format&lt;br&gt;[suffix] ::= (V</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:SCALe &lt;scale&gt;[suffix] (see page 183)</td>
<td>:CHANnel&lt;n&gt;:SCALe? (see page 183)</td>
<td>&lt;scale&gt; ::= Vertical units per division value in NR3 format&lt;br&gt;[suffix] ::= (V</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:UNITs &lt;units&gt; (see page 184)</td>
<td>:CHANnel&lt;n&gt;:UNITs? (see page 184)</td>
<td>&lt;units&gt; ::= {VOLTs</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:VERNier {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
</tbody>
</table>
### Table 8: :DIGital<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DIGital&lt;n&gt;:DISPlay ((0</td>
<td>OFF)</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:DIGital&lt;n&gt;:LABel</td>
<td>:DIGital&lt;n&gt;:LABel?</td>
<td>&lt;string&gt; ::= any series of 6 or less ASCII characters enclosed in quotation marks</td>
</tr>
<tr>
<td>&lt;string&gt; (see page 189)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:DIGital&lt;n&gt;:POSition</td>
<td>:DIGital&lt;n&gt;:POSition?</td>
<td></td>
</tr>
<tr>
<td>&lt;position&gt; (see page 190)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:DIGital&lt;n&gt;:SIZE</td>
<td>:DIGital&lt;n&gt;:SIZE?</td>
<td></td>
</tr>
<tr>
<td>&lt;value&gt; (see page 191)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:DIGital&lt;n&gt;:THReshold</td>
<td>:DIGital&lt;n&gt;:THReshold?</td>
<td></td>
</tr>
<tr>
<td>&lt;value&gt;[suffix] (see page 192)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Commands Quick Reference

### Table 9: :DISPlay Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DISPlay:CLEar (see page 195)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
| :DISPlay:DATA [<format>][,][<area>][,<] [<palette>] | :DISPlay:DATA? [<format>][,][<area>][,] [<palette>] (see page 196) | <format> ::= {TIFF} (command)  
<area> ::= {GRATicule} (command)  
<palette> ::= {MONochrome} (command)  
<format> ::= {TIFF | BMP | BMP8bit | PNG} (query)  
<area> ::= {GRATicule | SCReen} (query)  
<palette> ::= {MONochrome | GRAYscale | COlor} (query)  
<display data> ::= data in IEEE 488.2 # format |
| :DISPlay:LABel {{0 | OFF} | {1 | ON}} (see page 198) | :DISPlay:LABel? (see page 198) | {0 | 1} |
| :DISPlay:LABList <binary block> (see page 199) | :DISPlay:LABList? (see page 199) | <binary block> ::= an ordered list of up to 75 labels, each 6 characters maximum, separated by newline characters |
| :DISPlay:PERSistence <value> (see page 200) | :DISPlay:PERSistence? (see page 200) | <value> ::= {MINimum | INFinite}) |
| :DISPlay:SOURce <value> (see page 201) | :DISPlay:SOURce? (see page 201) | <value> ::= {PMEMory(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)} |
| :DISPlay:VECTors {{1 | ON} | {0 | OFF}} (see page 202) | :DISPlay:VECTors? (see page 202) | {1 | 0} |

### Table 10: :EXTernal Trigger Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:EXTernal:BWLimit &lt;bwlimit&gt; (see page 204)</td>
<td>:EXTernal:BWLimit? (see page 204)</td>
<td>&lt;bwlimit&gt; ::= {0</td>
</tr>
<tr>
<td>:EXTernal:IMPedance &lt;value&gt; (see page 205)</td>
<td>:EXTernal:IMPedance? (see page 205)</td>
<td>&lt;impedance&gt; ::= {ONEMeg</td>
</tr>
<tr>
<td>:EXTernal:PROBe &lt;attenuation&gt; (see page 206)</td>
<td>:EXTernal:PROBe? (see page 206)</td>
<td>&lt;attenuation&gt; ::= probe attenuation ratio in NR3 format</td>
</tr>
</tbody>
</table>
Table 10 :EXTernal Trigger Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a :EXTernal:PRoBE:ID? (see page 207)</td>
<td>&lt;probe id&gt; ::= unquoted ASCII string up to 11 characters</td>
<td></td>
</tr>
<tr>
<td>:EXTernal:PRoBE:STYPe &lt;signal type&gt; (see page 208) :EXTernal:PRoBE:STYPe? (see page 208)</td>
<td>&lt;signal type&gt; ::= (DIIfferential</td>
<td>SINGle)</td>
</tr>
<tr>
<td>:EXTernal:PRoTection[:CLEar] (see page 209) :EXTernal:PRoTection? (see page 209)</td>
<td>{NORM</td>
<td>TRIP}</td>
</tr>
<tr>
<td>:EXTernal:RANGE &lt;range&gt;[&lt;suffix&gt;] (see page 210) :EXTernal:RANGE? (see page 210)</td>
<td>&lt;range&gt; ::= vertical full-scale range value in NR3 format &lt;suffix&gt; ::= (V</td>
<td>mV)</td>
</tr>
<tr>
<td>:EXTernal:UNITs &lt;units&gt; (see page 211) :EXTernal:UNITs? (see page 211)</td>
<td>&lt;units&gt; ::= {VOLTs</td>
<td>AMPeres}</td>
</tr>
</tbody>
</table>

Table 11 :FUNCtion Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:FUNCtion:CENTer &lt;frequency&gt; (see page 214) :FUNCtion:CENTer? (see page 214)</td>
<td>&lt;frequency&gt; ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 25 GHz.</td>
<td></td>
</tr>
<tr>
<td>:FUNCtion:DISPlay {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:FUNCtion:OFFSet &lt;offset&gt; (see page 216) :FUNCtion:OFFSet? (see page 216)</td>
<td>&lt;offset&gt; ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function.</td>
<td></td>
</tr>
<tr>
<td>:FUNCtion:OPERation &lt;operation&gt; (see page 217) :FUNCtion:OPERation? (see page 217)</td>
<td>&lt;operation&gt; ::= {SUBTract</td>
<td>MULTiply</td>
</tr>
<tr>
<td>:FUNCtion:RANGE &lt;range&gt; (see page 218) :FUNCtion:RANGE? (see page 218)</td>
<td>&lt;range&gt; ::= 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 DIFFerentiate function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 11: :FUNCTION Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:FUNCTION:REFERENCE &lt;level&gt; (see page 219)</td>
<td>:FUNCTION:REFERENCE? (see page 219)</td>
<td>&lt;level&gt; ::= the current reference level in NR3 format. The range of legal values is from 400.0 dBV to +400.0 dBV (depending on current range value).</td>
</tr>
<tr>
<td>:FUNCTION:SCALE &lt;scale value&gt;[&lt;suffix&gt;] (see page 220)</td>
<td>:FUNCTION:SCALE? (see page 220)</td>
<td>&lt;scale value&gt; ::= integer in NR1 format &lt;suffix&gt; ::= (V</td>
</tr>
<tr>
<td>:FUNCTION:SOURCe &lt;source&gt; (see page 221)</td>
<td>:FUNCTION:SOURCe? (see page 221)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:FUNCTION:SPAN &lt;span&gt; (see page 222)</td>
<td>:FUNCTION:SPAN? (see page 222)</td>
<td>&lt;span&gt; ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.</td>
</tr>
<tr>
<td>:FUNCTION:WINDow &lt;window&gt; (see page 223)</td>
<td>:FUNCTION:WINDow? (see page 223)</td>
<td>&lt;window&gt; ::= (RECTangular</td>
</tr>
</tbody>
</table>

### Table 12: :HARDcopy Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:HARDcopy:FACTors {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:HARDcopy:FFEed {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:HARDcopy:FILename &lt;string&gt; (see page 227)</td>
<td>:HARDcopy:FILename? (see page 227)</td>
<td>&lt;string&gt; ::= quoted ASCII string</td>
</tr>
<tr>
<td>:HARDcopy:FORMat &lt;format&gt; (see page 228)</td>
<td>:HARDcopy:FORMat? (see page 228)</td>
<td>&lt;format&gt; ::= {BMP[24bit]</td>
</tr>
<tr>
<td>:HARDcopy:IGColors {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
</tbody>
</table>
### Table 12 :HARDcopy Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:HARDcopy:PALETTE &lt;palette&gt;</td>
<td>:HARDcopy:PALETTE?</td>
<td>&lt;palette&gt; ::= {COLOR</td>
</tr>
<tr>
<td>:HARDcopy:PDRIVER &lt;driver&gt;</td>
<td>:HARDcopy:PDRIVER?</td>
<td>&lt;driver&gt; ::= (AP2XX</td>
</tr>
</tbody>
</table>

### Table 13 :MARKer Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MARKer:MODE &lt;mode&gt;</td>
<td>:MARKer:MODE?</td>
<td>&lt;mode&gt; ::= {OFF</td>
</tr>
<tr>
<td>:MARKer:X1POSITION &lt;position&gt;[suffix]</td>
<td>:MARKer:X1POSITION?</td>
<td>&lt;position&gt; ::= X1 cursor position value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= {s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= X1 cursor position value in NR3 format</td>
</tr>
<tr>
<td>:MARKer:X1Y1SOURCE &lt;source&gt;</td>
<td>:MARKer:X1Y1SOURCE?</td>
<td>&lt;source&gt; ::= (CHANNEL&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= &lt;source&gt;</td>
</tr>
<tr>
<td>:MARKer:X2POSITION &lt;position&gt;[suffix]</td>
<td>:MARKer:X2POSITION?</td>
<td>&lt;position&gt; ::= X2 cursor position value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= {s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= X2 cursor position value in NR3 format</td>
</tr>
<tr>
<td>:MARKer:X2Y2SOURCE &lt;source&gt;</td>
<td>:MARKer:X2Y2SOURCE?</td>
<td>&lt;source&gt; ::= (CHANNEL&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= &lt;source&gt;</td>
</tr>
<tr>
<td>n/a</td>
<td>:MARKer:XDELTa?</td>
<td>&lt;return_value&gt; ::= X cursors delta value in NR3 format</td>
</tr>
</tbody>
</table>
### Table 13: MARKer Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MARKer:Y1Position &lt;position&gt;[suffix] (see page 240)</td>
<td>:MARKer:Y1Position? (see page 240)</td>
<td>&lt;position&gt; ::= Y1 cursor position value in NR3 format&lt;br&gt;[suffix] ::= (V</td>
</tr>
<tr>
<td>:MARKer:Y2Position &lt;position&gt;[suffix] (see page 241)</td>
<td>:MARKer:Y2Position? (see page 241)</td>
<td>&lt;position&gt; ::= Y2 cursor position value in NR3 format&lt;br&gt;[suffix] ::= (V</td>
</tr>
<tr>
<td>N/A</td>
<td>:MARKer:YDELta? (see page 242)</td>
<td>&lt;return_value&gt; ::= Y cursors delta value in NR3 format</td>
</tr>
</tbody>
</table>

### Table 14: MEASure Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASure:CLEAR (see page 250)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>:MEASure:COUNTER [&lt;source&gt;] (see page 251)</td>
<td>:MEASure:COUNTER? [&lt;source&gt;] (see page 251)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;) for DSO models&lt;br&gt;&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:DEFINE DELay, &lt;delay spec&gt; (see page 252)</td>
<td>:MEASure:DEFINE? DELay (see page 253)</td>
<td>&lt;delay spec&gt; ::= &lt;edge_spec&gt;,&lt;edge_spec2&gt;&lt;edge_spec1 ::= [&lt;slope&gt;]&lt;occurrence&gt;&lt;edge_spec2 ::= [&lt;slope&gt;]&lt;occurrence&gt;&lt;slope&gt; ::= (+</td>
</tr>
<tr>
<td>:MEASure:DEFINE THResholds, &lt;threshold spec&gt; (see page 252)</td>
<td>:MEASure:DEFINE? THResholds (see page 253)</td>
<td>&lt;threshold spec&gt; ::= (STANdard)</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>:MEASure:DELay</td>
<td>:MEASure:DELay?</td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source1&gt;]</td>
<td>[&lt;source1&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>[,&lt;source2&gt;]</td>
<td>[,&lt;source2&gt;]</td>
<td>&lt;return_value&gt; ::= floating-point number delay time in seconds in NR3 format</td>
</tr>
<tr>
<td>(see page 255)</td>
<td>(see page 255)</td>
<td></td>
</tr>
<tr>
<td>:MEASure:DUTYcycle</td>
<td>:MEASure:DUTYcycle?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 257)</td>
<td>(see page 257)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FALLtime</td>
<td>:MEASure:FALLtime?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 258)</td>
<td>(see page 258)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FREQuency</td>
<td>:MEASure:FREQuency?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 259)</td>
<td>(see page 259)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:NWIDth</td>
<td>:MEASure:NWIDth?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 260)</td>
<td>(see page 260)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FALltime</td>
<td>:MEASure:FALltime?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 258)</td>
<td>(see page 258)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FREQuency</td>
<td>:MEASure:FREQuency?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 259)</td>
<td>(see page 259)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:NWIDth</td>
<td>:MEASure:NWIDth?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 260)</td>
<td>(see page 260)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FALltime</td>
<td>:MEASure:FALltime?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 258)</td>
<td>(see page 258)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FREQuency</td>
<td>:MEASure:FREQuency?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 259)</td>
<td>(see page 259)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:NWIDth</td>
<td>:MEASure:NWIDth?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 260)</td>
<td>(see page 260)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FALltime</td>
<td>:MEASure:FALltime?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 258)</td>
<td>(see page 258)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:MEASure:FREQuency</td>
<td>:MEASure:FREQuency?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 259)</td>
<td>(see page 259)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>
Table 14  :MEASURE Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :MEASURE:OVERshoot  | :MEASURE:OVERshoot?     | `<source> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= the percent of the overshoot of the selected waveform in NR3 format  
| [<source>]          | [<source>]              |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 261)          |                                                                                                                                                                                                                                                                                                                                                                                  |
| :MEASURE:PERiod     | :MEASURE:PERiod?        | `<source> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= waveform period in seconds in NR3 format  
| [<source>]          | [<source>]              |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 263)          |                                                                                                                                                                                                                                                                                                                                                                                  |
| :MEASURE:PHASe       | :MEASURE:PHASe?         | `<source1,2> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= the phase angle value in degrees in NR3 format  
| [<source1>]         | [<source1>]             |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | [,<source2>]             |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 264)          |                                                                                                                                                                                                                                                                                                                                                                                  |
| :MEASURE:PREShoot   | :MEASURE:PREShoot?      | `<source> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= the percent of preshoot of the selected waveform in NR3 format  
| [<source>]          | [<source>]              |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 265)          |                                                                                                                                                                                                                                                                                                                                                                                  |
| :MEASURE:PWIDth      | :MEASURE:PWIDth?        | `<source> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= width of positive pulse in seconds in NR3 format  
| [<source>]          | [<source>]              |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 266)          |                                                                                                                                                                                                                                                                                                                                                                                  |
| :MEASURE:RISEtime    | :MEASURE:RISEtime?      | `<source> ::= (CHANnel<n> | FUNCTION | MATH)`  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= rise time in seconds in NR3 format  
<p>| [&lt;source&gt;]          | [&lt;source&gt;]              |                                                                                                                                                                                                                                                                                                                                                                                  |
|                      | (see page 267)          |                                                                                                                                                                                                                                                                                                                                                                                  |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASURE:SDEViation [&lt;source&gt;] (see page 268)</td>
<td>:MEASURE:SDEViation? [&lt;source&gt;] (see page 268)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASURE:SHOW (1</td>
<td>ON) (see page 269)</td>
<td>:MEASURE:SHOW? (see page 269)</td>
</tr>
<tr>
<td>:MEASURE:SOURce [&lt;source1&gt;] [,&lt;source2&gt;] (see page 270)</td>
<td>:MEASURE:SOURce? (see page 270)</td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>n/a</td>
<td>:MEASURE:TEDGe? &lt;slope&gt;&lt;occurrence&gt;[,&lt;source&gt;] (see page 272)</td>
<td>&lt;slope&gt; ::= direction of the waveform &lt;occurrence&gt; ::= the transition to be reported &lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
</tbody>
</table>
### Table 14: :MEASure Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:MEASure:TVALue?</td>
<td>&lt;value&gt; ::= voltage level that the waveform must cross.</td>
</tr>
<tr>
<td></td>
<td>[,&lt;slope&gt;,&lt;occurrence&gt;]</td>
<td>&lt;slope&gt; ::= direction of the waveform when &lt;value&gt; is crossed.</td>
</tr>
<tr>
<td></td>
<td>[],&lt;source&gt;]</td>
<td>&lt;occurrence&gt; ::= transitions reported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= time in seconds of specified voltage crossing in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:VAMPlitude</td>
<td>:MEASure:VAMPlitude?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[,&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= the amplitude of the selected waveform in volts in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VAVerage</td>
<td>:MEASure:VAVerage?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[,&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= calculated average voltage in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VBASE</td>
<td>:MEASure:VBASE?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[,&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;base_voltage&gt; ::= voltage at the base of the selected waveform in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VMAX</td>
<td>:MEASure:VMAX?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[,&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= maximum voltage of the selected waveform in NR3 format</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>:MEASure:VMIN</td>
<td>:MEASure:VMIN?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>(see page 280)</td>
<td>(see page 280)</td>
<td>&lt;return_value&gt; ::= minimum voltage of the selected waveform in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VPP</td>
<td>:MEASure:VPP?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>(see page 281)</td>
<td>(see page 281)</td>
<td>&lt;return_value&gt; ::= voltage peak-to-peak of the selected waveform in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VRMS</td>
<td>:MEASure:VRMS?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>(see page 282)</td>
<td>(see page 282)</td>
<td>&lt;return_value&gt; ::= calculated dc RMS voltage in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:MEASure:VTIME?</td>
<td>&lt;vtime&gt; ::= displayed time from trigger in seconds in NR3 format</td>
</tr>
<tr>
<td>&lt;vtime&gt;,[&lt;source&gt;]</td>
<td>(see page 283)</td>
<td>&lt;return_value&gt; ::= voltage at the specified time in NR3 format</td>
</tr>
<tr>
<td>:MEASure:VTOP</td>
<td>:MEASure:VTOP?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>[&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 284)</td>
<td>(see page 284)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= voltage at the top of the waveform in NR3 format</td>
</tr>
</tbody>
</table>
### Table 14: :MEASure Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASure:XMAX</td>
<td>:MEASure:XMAX?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td>([&lt;source&gt;] (see</td>
<td>&lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</td>
</tr>
<tr>
<td></td>
<td>page 285)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:MEASure:XMIN</td>
<td>:MEASure:XMIN?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td>([&lt;source&gt;] (see</td>
<td>&lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</td>
</tr>
<tr>
<td></td>
<td>page 286)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 15: :POD<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:POD&lt;n&gt;:DISPlay</td>
<td>:POD&lt;n&gt;:DISPlay?</td>
<td>(0</td>
</tr>
<tr>
<td>[{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:POD&lt;n&gt;:SIZE</td>
<td>:POD&lt;n&gt;:SIZE?</td>
<td>&lt;value&gt; ::= (SMALl</td>
</tr>
<tr>
<td>&lt;value&gt;</td>
<td>(see page 289)</td>
<td></td>
</tr>
<tr>
<td>:POD&lt;n&gt;:THReshold</td>
<td>:POD&lt;n&gt;:THReshold?</td>
<td>&lt;n&gt; ::= 1-2 in NR1 format</td>
</tr>
<tr>
<td>&lt;type&gt;[suffix]</td>
<td>(see page 290)</td>
<td>&lt;type&gt; ::= {CMOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;user defined value&gt; ::= value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= {V</td>
</tr>
</tbody>
</table>

### Table 16: :SBUS Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:BUSDoctor:ADDRESS &lt;value&gt; (see page 294)</td>
<td>:SBUS:BUSDoctor:ADDRESS? (see page 294)</td>
<td>&lt;value&gt; ::= &lt;field value&gt;, &lt;field value&gt;, &lt;field value&gt;, &lt;field value&gt; in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;field value&gt; ::= integer from 0-255</td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:BAUDrate &lt;baudrate&gt; (see page 295)</td>
<td>:SBUS:BUSDoctor:BAUDrate? (see page 295)</td>
<td>&lt;baudrate&gt; ::= (2500000</td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:CHANnel &lt;channel&gt; (see page 296)</td>
<td>:SBUS:BUSDoctor:CHANnel? (see page 296)</td>
<td>&lt;channel&gt; ::= (A</td>
</tr>
</tbody>
</table>

---

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

**Table 15:** :POD<n> Commands Summary

**Table 16:** :SBUS Commands Summary
### Table 16: SBUS Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:BUSDoctor:MODE &lt;mode&gt; (see page 297)</td>
<td>:SBUS:BUSDoctor:MODE? (see page 297)</td>
<td>&lt;mode&gt; ::= (ASYNchronous</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:ERROR? (see page 298)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:OVERload? (see page 299)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNT:RESET (see page 300)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:TOTAL? (see page 301)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:UTILization? (see page 302)</td>
<td>&lt;percent&gt; ::= floating-point in NRI format</td>
</tr>
<tr>
<td>:SBUS:DISPLAY {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:NULL? (see page 304)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>:SBUS:FLEXray:COUNT:RESET (see page 305)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:SYNC? (see page 306)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:TOTAL? (see page 307)</td>
<td>&lt;frame_count&gt; ::= integer in NRI format</td>
</tr>
<tr>
<td>:SBUS:IIC:SIZE &lt;size&gt; (see page 308)</td>
<td>:SBUS:IIC:SIZE? (see page 308)</td>
<td>&lt;size&gt; ::= (BIT7</td>
</tr>
<tr>
<td>:SBUS:LIN:PARity {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:SBUS:MODE &lt;mode&gt; (see page 310)</td>
<td>:SBUS:MODE? (see page 310)</td>
<td>&lt;mode&gt; ::= (IIC</td>
</tr>
<tr>
<td>:SBUS:SPI:WIDTH &lt;word_width&gt; (see page 311)</td>
<td>:SBUS:SPI:WIDTH? (see page 311)</td>
<td>&lt;word_width&gt; ::= integer 4-16 in NRI format</td>
</tr>
</tbody>
</table>
## Commands Quick Reference

### Table 17 :SYSTem Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :SYSTem:DATE <date> (see page 313) | :SYSTem:DATE? (see page 313) | <date> ::= <year>,<month>,<day>  
<year> ::= 4-digit year in NR1 format  
<month> ::= (1,...,12 | JANuary | FEBruary | MARch | APRil | MAY | JUNE | JULy | AUGust | SEPtember | OCTober | NOVember | DECember)  
<day> ::= (1,...31) |
| :SYSTem:DSP <string> (see page 314) | n/a | <string> ::= up to 254 characters as a quoted ASCII string |
| n/a | :SYSTem:ERRor? (see page 315) | <error> ::= an integer error code  
<error string> ::= quoted ASCII string.  
See Error Messages (see page 533). |
| :SYSTem:LOCK (see page 316) | :SYSTem:LOCK? (see page 316) | <value> ::= (ON | OFF) |
| :SYSTem:SETup <setup_data> (see page 317) | :SYSTem:SETup? (see page 317) | <setup_data> ::= data in IEEE 488.2 # format. |
| :SYSTem:TIME <time> (see page 319) | :SYSTem:TIME? (see page 319) | <time> ::= hours,minutes,seconds in NR1 format |

### Table 18 :TIMebase Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TIMebase:MODE &lt;value&gt; (see page 322)</td>
<td>:TIMebase:MODE? (see page 322)</td>
<td>&lt;value&gt; ::= (MAIN</td>
</tr>
<tr>
<td>:TIMebase:POSition &lt;pos&gt; (see page 323)</td>
<td>:TIMebase:POSition? (see page 323)</td>
<td>&lt;pos&gt; ::= time from the trigger event to the display reference point in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:RANGe &lt;range_value&gt; (see page 324)</td>
<td>:TIMebase:RANGe? (see page 324)</td>
<td>&lt;range_value&gt; ::= 5 ns through 500 s in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:REFClock {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:TIMebase:REFerence {LEFT</td>
<td>CENTer</td>
<td>RIGHT} (see page 326)</td>
</tr>
</tbody>
</table>
Table 18  :TIMebase Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TIMebase:SCALE &lt;scale_value&gt; (see page 327)</td>
<td>:TIMebase:SCALE? (see page 327)</td>
<td>&lt;scale_value&gt; ::= scale value in seconds in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:VERNier {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:TIMebase:WINDow:POSITION &lt;pos&gt; (see page 329)</td>
<td>:TIMebase:WINDow:POSITION? (see page 329)</td>
<td>&lt;pos&gt; ::= time from the trigger event to the delayed view reference point in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:WINDow:RANGE &lt;range_value&gt; (see page 330)</td>
<td>:TIMebase:WINDow:RANGE? (see page 330)</td>
<td>&lt;range value&gt; ::= range value in seconds in NR3 format for the delayed window</td>
</tr>
<tr>
<td>:TIMebase:WINDow:SCALE &lt;scale_value&gt; (see page 331)</td>
<td>:TIMebase:WINDow:SCALE? (see page 331)</td>
<td>&lt;scale_value&gt; ::= scale value in seconds in NR3 format for the delayed window</td>
</tr>
</tbody>
</table>

Table 19  General :TRIGger Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:HFReject {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:TRIGger:HOLDoff &lt;holdoff_time&gt; (see page 337)</td>
<td>:TRIGger:HOLDoff? (see page 337)</td>
<td>&lt;holdoff_time&gt; ::= 60 ns to 10 s in NR3 format</td>
</tr>
<tr>
<td>:TRIGger:MODE &lt;mode&gt; (see page 338)</td>
<td>:TRIGger:MODE? (see page 338)</td>
<td>&lt;mode&gt; ::= {EDGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= {&lt;mode&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;none&gt; ::= query returns &quot;NONE&quot; if the :TIMebase:MODE is ROLL or XY</td>
</tr>
<tr>
<td>:TRIGger:NREJect {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
</tbody>
</table>
### Table 19  General :TRIGger Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :TRIGger:PATTern <value>, <mask> [,<edge source>,<edge>] (see page 340) | :TRIGger:PATTern? (see page 341) | <value> ::= 32-bit integer or <string>  
<mask> ::= 32-bit integer or <string>  
<string> ::= "0xnnnnnn"; n ::= (0,...,9 | A,...,F)  
<edge source> ::= (CHANnel<n> | EXTernal | NONE) for DSO models  
<edge source> ::= (CHANnel<n> | DIGital0,...,DIGital15 | NONE) for MSO models  
<edge> ::= {POSitive | NEGative}  
<n> ::= 1-2 or 1-4 in NR1 format |
| :TRIGger:SWEep <sweep> (see page 342) | :TRIGger:SWEep? (see page 342) | <sweep> ::= {AUTO | NORMal} |

### Table 20  :TRIGger:CAN Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :TRIGger:CAN:PATTern:DATA <value>, <mask> (see page 345) | :TRIGger:CAN:PATTern:DATA? (see page 345) | <value> ::= 64-bit integer in decimal, <nondecimal>, or <string> (with Option AMS)  
<mask> ::= 64-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 |
| :TRIGger:CAN:PATTern:DATA:LENGTH <length> (see page 346) | :TRIGger:CAN:PATTern:DATA:LENGTH? (see page 346) | <length> ::= integer from 1 to 8 in NR1 format (with Option AMS) |
Table 20  :TRIGger:CAN Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:CAN:PATTern:ID &lt;value&gt;, &lt;mask&gt;</td>
<td>:TRIGger:CAN:PATTern:ID ?</td>
<td>&lt;value&gt; ::= 32-bit integer in decimal, &lt;nondecimal&gt;, or &lt;string&gt; (with Option AMS)</td>
</tr>
<tr>
<td></td>
<td>(see page 347)</td>
<td>&lt;mask&gt; ::= 32-bit integer in decimal, &lt;nondecimal&gt;, or &lt;string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;nondecimal&gt; ::= #Hnn...n where n ::= {0,..,9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;nondecimal&gt; ::= #Bnn...n where n ::= {0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;string&gt; ::= &quot;0xnn...n&quot; where n ::= {0,..,9</td>
</tr>
<tr>
<td></td>
<td>(see page 348)</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:CAN:SAMPlepoint &lt;value&gt;</td>
<td>:TRIGger:CAN:SAMPlepoint?</td>
<td>&lt;value&gt; ::= (60</td>
</tr>
<tr>
<td></td>
<td>(see page 349)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(see page 350)</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:CAN:SOURce &lt;source&gt;</td>
<td>:TRIGger:CAN:SOURce?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td>(see page 351)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td>(see page 352)</td>
<td>&lt;condition&gt; ::= (SOF</td>
</tr>
</tbody>
</table>
## Table 21 :TRIGger:DURation Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:DURation:GREater than &lt;greater than time&gt;[suffix] (see page 355)</td>
<td>:TRIGger:DURation:GREater than? (see page 355)</td>
<td>&lt;greater than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:DURation:LESSthan &lt;less than time&gt;[suffix] (see page 356)</td>
<td>:TRIGger:DURation:LESSthan? (see page 356)</td>
<td>&lt;less than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:DURation:PATTERN &lt;value&gt;, &lt;mask&gt; (see page 357)</td>
<td>:TRIGger:DURation:PATTERN? (see page 357)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;mask&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;**0xnnnnnn&quot; n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:DURation:QUALifier &lt;qualifier&gt; (see page 358)</td>
<td>:TRIGger:DURation:QUALifier? (see page 358)</td>
<td>&lt;qualifier&gt; ::= (GREaterthan</td>
</tr>
<tr>
<td>:TRIGger:DURation:RANGE &lt;greater than time&gt;[suffix], &lt;less than time&gt;[suffix] (see page 359)</td>
<td>:TRIGger:DURation:RANGE? (see page 359)</td>
<td>&lt;greater than time&gt; ::= min duration from 10 ns to 9.99 seconds in NR3 format &lt;less than time&gt; ::= max duration from 15 ns to 10 seconds in NR3 format [suffix] ::= (s</td>
</tr>
</tbody>
</table>

## Table 22 :TRIGger:EBURst Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:EBURst:COUNT &lt;count&gt; (see page 361)</td>
<td>:TRIGger:EBURst:COUNT? (see page 361)</td>
<td>&lt;count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:EBURst:IDLE &lt;time_value&gt; (see page 362)</td>
<td>:TRIGger:EBURst:IDLE? (see page 362)</td>
<td>&lt;time_value&gt; ::= time in seconds in NR3 format</td>
</tr>
<tr>
<td>:TRIGger:EBURst:SLOPe &lt;slope&gt; (see page 363)</td>
<td>:TRIGger:EBURst:SLOPe? (see page 363)</td>
<td>&lt;slope&gt; ::= (NEGative</td>
</tr>
</tbody>
</table>
### Table 23: :TRIGger[:EDGE] Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger[:EDGE]:COUPling {AC</td>
<td>DC</td>
<td>LF} (see page 365)</td>
</tr>
<tr>
<td>:TRIGger[:EDGE]:LEVel &lt;level&gt; [,&lt;source&gt;] (see page 366)</td>
<td>:TRIGger[:EDGE]:LEVel? [&lt;source&gt;] (see page 366)</td>
<td>For internal triggers, &lt;level&gt; ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers, &lt;level&gt; ::= 2 volts with probe attenuation at 1:1 in NR3 format. For digital channels (MSO models), &lt;level&gt; ::= 8 V. &lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger[:EDGE]:REJect {OFF</td>
<td>LF</td>
<td>HF} (see page 367)</td>
</tr>
<tr>
<td>:TRIGger[:EDGE]:SLOPe &lt;polarity&gt; (see page 368)</td>
<td>:TRIGger[:EDGE]:SLOPe? (see page 368)</td>
<td>&lt;polarity&gt; ::= {POSitive</td>
</tr>
<tr>
<td>:TRIGger[:EDGE]:SOURce &lt;source&gt; (see page 369)</td>
<td>:TRIGger[:EDGE]:SOURce? (see page 369)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
</tbody>
</table>

### Table 24: :TRIGger:FLEXray Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:FLEXray:ERRor:TYPE &lt;error_type&gt; (see page 371)</td>
<td>:TRIGger:FLEXray:ERRor:TYPE? (see page 371)</td>
<td>&lt;error_type&gt; ::= {ALL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:CCBase &lt;cycle_count_base&gt; (see page 373)</td>
<td>:TRIGger:FLEXray:FRAME:CCBase? (see page 373)</td>
<td>&lt;cycle_count_base&gt; ::= integer from 0-63</td>
</tr>
</tbody>
</table>
### Table 24: :TRIGger:FLEXray Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:FLEXray:FRAME:CCRepetition &lt;cycle_count_repetition&gt; (see page 374)</td>
<td>:TRIGger:FLEXray:FRAME:CCRepetition? (see page 374)</td>
<td>&lt;cycle_count_repetition&gt; ::= (ALL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:ID &lt;frame_id&gt; (see page 375)</td>
<td>:TRIGger:FLEXray:FRAME:ID? (see page 375)</td>
<td>&lt;frame_id&gt; ::= (ALL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:FRAME:TYPE &lt;frame_type&gt; (see page 376)</td>
<td>:TRIGger:FLEXray:FRAME:TYPE? (see page 376)</td>
<td>&lt;frame_type&gt; ::= (NORMAL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:CBASE &lt;cycle_base&gt; (see page 377)</td>
<td>:TRIGger:FLEXray:TIME:CBASE? (see page 377)</td>
<td>&lt;cycle_base&gt; ::= integer from 0-63</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:CRepetition &lt;cycle_repetition&gt; (see page 378)</td>
<td>:TRIGger:FLEXray:TIME:CRepetition? (see page 378)</td>
<td>&lt;cycle_repetition&gt; ::= (ALL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:SEGment &lt;segment_type&gt; (see page 379)</td>
<td>:TRIGger:FLEXray:TIME:SEGment? (see page 379)</td>
<td>&lt;segment_type&gt; ::= (STATic</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TIME:SLOT &lt;slot_type&gt;, &lt;slot_id&gt; (see page 380)</td>
<td>:TRIGger:FLEXray:TIME:SLOT? (see page 380)</td>
<td>&lt;slot_type&gt; ::= (ALL</td>
</tr>
<tr>
<td>:TRIGger:FLEXray:TRIGGER &lt;condition&gt; (see page 381)</td>
<td>:TRIGger:FLEXray:TRIGGER? (see page 381)</td>
<td>&lt;condition&gt; ::= (FRAME</td>
</tr>
</tbody>
</table>

### Table 25: :TRIGger:GLITch Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:GLITch:GREater than &lt;greater than time&gt;[suffix] (see page 384)</td>
<td>:TRIGger:GLITch:GREater than? (see page 384)</td>
<td>&lt;greater than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:GLITch:LESSthan &lt;less than time&gt;[suffix] (see page 385)</td>
<td>:TRIGger:GLITch:LESSthan? (see page 385)</td>
<td>&lt;less than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>:TRIGger:GLITch:LEVEL &lt;level&gt; [source] (see page 386)</td>
<td>:TRIGger:GLITch:LEVEL? (see page 386)</td>
<td>For internal triggers, &lt;level&gt; := .75 x full-scale voltage from center screen in NR3 format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For external triggers, &lt;level&gt; := 2 volts with probe attenuation at 1:1 in NR3 format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For digital channels (MSO models), &lt;level&gt; := 6 V.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:GLITch:POLarity &lt;polarity&gt; (see page 387)</td>
<td>:TRIGger:GLITch:POLarity? (see page 387)</td>
<td>&lt;polarity&gt; ::= (POSitive</td>
</tr>
<tr>
<td>:TRIGger:GLITch:QUALifier &lt;qualifier&gt; (see page 388)</td>
<td>:TRIGger:GLITch:QUALifier? (see page 388)</td>
<td>&lt;qualifier&gt; ::= (GREaterthan</td>
</tr>
<tr>
<td>:TRIGger:GLITch:RANGe &lt;greater than time&gt;[suffix], &lt;less than time&gt;[suffix] (see page 389)</td>
<td>:TRIGger:GLITch:RANGe? (see page 389)</td>
<td>&lt;greater than time&gt; ::= start time from 10 ns to 9.99 seconds in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;less than time&gt; ::= stop time from 15 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:GLITch:SOURce &lt;source&gt; (see page 390)</td>
<td>:TRIGger:GLITch:SOURce? (see page 390)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>:TRIGger:IIC:PA&lt;Tattern:ADRESS &lt;value&gt; (see page 392)</td>
<td>:TRIGger:IIC:PA&lt;Ttern:ADRESS? (see page 392)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnn&quot; n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:IIC:PA&lt;Ttern:DA&lt;T &lt;value&gt; (see page 393)</td>
<td>:TRIGger:IIC:PA&lt;Ttern:DA&lt;T? (see page 393)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnn&quot; n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:IIC:PA&lt;Ttern:DA&lt;Ter &lt;value&gt; (see page 394)</td>
<td>:TRIGger:IIC:PA&lt;Ttern:DA&lt;Ter? (see page 394)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnn&quot; n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:IIC[:SOURce]:CLock &lt;source&gt; (see page 395)</td>
<td>:TRIGger:IIC[:SOURce]:CLock? (see page 395)</td>
<td>&lt;source&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:IIC[:SOURce]:DATA &lt;source&gt; (see page 396)</td>
<td>:TRIGger:IIC[:SOURce]:DATA? (see page 396)</td>
<td>&lt;source&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:IIC:TRIGger:QUALifier &lt;value&gt; (see page 397)</td>
<td>:TRIGger:IIC:TRIGger:QUALifier? (see page 397)</td>
<td>&lt;value&gt; ::= {EQUal</td>
</tr>
<tr>
<td>:TRIGger:IIC:TRIGger[:TYPe] &lt;type&gt; (see page 398)</td>
<td>:TRIGger:IIC:TRIGger[:TYPe]? (see page 398)</td>
<td>&lt;value&gt; ::= {STARt</td>
</tr>
</tbody>
</table>
### Table 27 :TRIGger:LIN Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:LIN:ID &lt;value&gt; (see page 401)</td>
<td>:TRIGger:LIN:ID? (see page 401)</td>
<td>&lt;value&gt; ::= 7-bit integer in decimal, &lt;nondecimal&gt;, or &lt;string&gt; from 0-63 or 0x00-0x3f (with Option AMS) &lt;nondecimal&gt; ::= #Hnn where n ::= (0,..,9</td>
</tr>
<tr>
<td>:TRIGger:LIN:SAMPLEpoint &lt;value&gt; (see page 402)</td>
<td>:TRIGger:LIN:SAMPLEpoint? (see page 402)</td>
<td>&lt;value&gt; ::= {60</td>
</tr>
<tr>
<td>:TRIGger:LIN:SIGNal:BAUDrate &lt;baudrate&gt; (see page 403)</td>
<td>:TRIGger:LIN:SIGNal:BAUDrate? (see page 403)</td>
<td>&lt;baudrate&gt; ::= {2400</td>
</tr>
<tr>
<td>:TRIGger:LIN:SOURce &lt;source&gt; (see page 404)</td>
<td>:TRIGger:LIN:SOURce? (see page 404)</td>
<td>&lt;source&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:LIN:STANDard &lt;std&gt; (see page 405)</td>
<td>:TRIGger:LIN:STANDard? (see page 405)</td>
<td>&lt;std&gt; ::= {LIN13</td>
</tr>
<tr>
<td>:TRIGger:LIN:SYNCbreak &lt;value&gt; (see page 406)</td>
<td>:TRIGger:LIN:SYNCbreak? (see page 406)</td>
<td>&lt;value&gt; ::= integer = (11</td>
</tr>
<tr>
<td>:TRIGger:LIN:TRIGger &lt;condition&gt; (see page 407)</td>
<td>:TRIGger:LIN:TRIGger? (see page 407)</td>
<td>&lt;condition&gt; ::= {SYNCbreak} (without Option AMS) &lt;condition&gt; ::= {SYNCbreak</td>
</tr>
</tbody>
</table>
### Table 28: TRIGGER:SEQUence Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGGER:SEQUence:COUNt &lt;count&gt; (see page 409)</td>
<td>:TRIGGER:SEQUence:COUNt ? (see page 409)</td>
<td>&lt;count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:EDGE{1</td>
<td>2} &lt;source&gt;, &lt;slope&gt; (see page 410)</td>
<td>:TRIGGER:SEQUence:EDGE{1</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:FIND &lt;value&gt; (see page 411)</td>
<td>:TRIGGER:SEQUence:FIND? (see page 411)</td>
<td>&lt;value&gt; ::= {PATTern1,ENTered</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:PATTERN{1</td>
<td>2} &lt;value&gt;, &lt;mask&gt; (see page 412)</td>
<td>:TRIGGER:SEQUence:PATTern{1</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:RESet &lt;value&gt; (see page 413)</td>
<td>:TRIGGER:SEQUence:RESet ? (see page 413)</td>
<td>&lt;value&gt; ::= {NONE</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:TIMer &lt;time_value&gt; (see page 414)</td>
<td>:TRIGGER:SEQUence:TIMer ? (see page 414)</td>
<td>&lt;time_value&gt; ::= time from 100 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td>:TRIGGER:SEQUence:TRIGGER &lt;value&gt; (see page 415)</td>
<td>:TRIGGER:SEQUence:TRIGGER? (see page 415)</td>
<td>&lt;value&gt; ::= {PATTern2,ENTered</td>
</tr>
</tbody>
</table>
# Table 29: :TRIGger:SPI Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:SPI:CLOCK:SLOPe &lt;slope&gt; (see page 417)</td>
<td>:TRIGger:SPI:CLOCK:SLOPe? (see page 417)</td>
<td>&lt;slope&gt; ::= {NEGative</td>
</tr>
<tr>
<td>:TRIGger:SPI:CLOCK:TIMEout &lt;time_value&gt; (see page 418)</td>
<td>:TRIGger:SPI:CLOCK:TIMEout? (see page 418)</td>
<td>&lt;time_value&gt; ::= time in seconds in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:SPI:FRAMing &lt;value&gt; (see page 419)</td>
<td>:TRIGger:SPI:FRAMing? (see page 419)</td>
<td>&lt;value&gt; ::= {CHIPselect</td>
</tr>
<tr>
<td>:TRIGger:SPI:PATTern:DATA &lt;value&gt;, &lt;mask&gt; (see page 420)</td>
<td>:TRIGger:SPI:PATTern:DATA? (see page 420)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;mask&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnnnnnn&quot; where n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:SPI:PATTern:WIDTh &lt;width&gt; (see page 421)</td>
<td>:TRIGger:SPI:PATTern:WIDTh? (see page 421)</td>
<td>&lt;width&gt; ::= integer from 4 to 32 in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:CLOCK &lt;source&gt; (see page 422)</td>
<td>:TRIGger:SPI:SOURce:CLOCK? (see page 422)</td>
<td>&lt;value&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:DATA &lt;source&gt; (see page 423)</td>
<td>:TRIGger:SPI:SOURce:DATA? (see page 423)</td>
<td>&lt;value&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:FRAME &lt;source&gt; (see page 424)</td>
<td>:TRIGger:SPI:SOURce:FRAME? (see page 424)</td>
<td>&lt;value&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
</tbody>
</table>
## 2 Commands Quick Reference

### Table 30 :TRIGger:TV Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:TV:LINE &lt;line number&gt; (see page 426)</td>
<td>:TRIGger:TV:LINE? (see page 426)</td>
<td>&lt;line number&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:TV:MODE &lt;tv mode&gt; (see page 427)</td>
<td>:TRIGger:TV:MODE? (see page 427)</td>
<td>&lt;tv mode&gt; ::= (FIEld1</td>
</tr>
<tr>
<td>:TRIGger:TV:POLarity &lt;polarity&gt; (see page 428)</td>
<td>:TRIGger:TV:POLarity? (see page 428)</td>
<td>&lt;polarity&gt; ::= (POSitive</td>
</tr>
<tr>
<td>:TRIGger:TV:SOURce &lt;source&gt; (see page 429)</td>
<td>:TRIGger:TV:SOURce? (see page 429)</td>
<td>&lt;source&gt; ::= {CHANnel&lt;n&gt;}</td>
</tr>
<tr>
<td>:TRIGger:TV:STANdard &lt;standard&gt; (see page 430)</td>
<td>:TRIGger:TV:STANdard? (see page 430)</td>
<td>&lt;standard&gt; ::= (GENeric</td>
</tr>
</tbody>
</table>

### Table 31 :TRIGger:USB Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:USB:SOURce:DMI Nus &lt;source&gt; (see page 432)</td>
<td>:TRIGger:USB:SOURce:DMI Nus? (see page 432)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:USB:SOURce:DPL us &lt;source&gt; (see page 433)</td>
<td>:TRIGger:USB:SOURce:DPL us? (see page 433)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:USB:SPEed &lt;value&gt; (see page 434)</td>
<td>:TRIGger:USB:SPEed? (see page 434)</td>
<td>&lt;value&gt; ::= (LOW</td>
</tr>
<tr>
<td>:TRIGger:USB:TRIGger &lt;value&gt; (see page 435)</td>
<td>:TRIGger:USB:TRIGger? (see page 435)</td>
<td>&lt;value&gt; ::= (SOP</td>
</tr>
</tbody>
</table>
### Table 32: :WAVeform Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:WAVeform:BYTEorder</td>
<td>:WAVeform:BYTEorder?</td>
<td>&lt;value&gt; ::= (LSBFirst</td>
</tr>
<tr>
<td>&lt;value&gt; (see page 444)</td>
<td>(see page 444)</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:COUNt?</td>
<td>&lt;count&gt; ::= an integer from 1 to 65536 in NR1 format</td>
</tr>
<tr>
<td></td>
<td>(see page 445)</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:DATA?</td>
<td>&lt;binary block length bytes&gt;, &lt;binary data&gt;</td>
</tr>
</tbody>
</table>
|                           | (see page 446)               | 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         | :WAVeform:FORMat?            | <value> ::= (WORD | BYTE | ASCII)                                |
| <value> (see page 448)    | (see page 448)               |                                                                  |
| :WAVeform:POINts         | :WAVeform:POINts?            | <# points> ::= (100 | 250 | 500 | 1000 | <points_mode>) if waveform points mode is NORMal |
| <# points> (see page 449) | (see page 449)               |                                                                  |
|                           |                              | <# points> ::= (100 | 250 | 500 | 1000 | 2000 ... 8000000 in 1-2-5 sequence | <points_mode>) if waveform points mode is MAXimum or RAW |
|                           |                              |                                                                  |
| <points_mode> (see page 451) | (see page 452)              |                                                                  |
### Table 32: :WAVeform Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| n/a     | :WAVeform:PREamble? (see page 453) | <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  
• 2 for AVERAGE type  
• 3 for HRESolution type  
<count> ::= Average count, or 1 if PEAK detect type or NORMAL; an integer in NR1 format |
| :WAVeform:SOURce <source> (see page 456) | :WAVeform:SOURce? (see page 456) | <source> ::= (CHANnel<n> | FUNCTION | MATH | SBUS) for DSO models  
<source> ::= (CHANnel<n> | POD(1 | 2) | BUS(1 | 2) | FUNCTION | MATH | SBUS) for MSO models  
<n> ::= 1-2 or 1-4 in NR1 format |
| n/a     | :WAVeform:TYPE? (see page 460) |  
<return_mode> ::= (NORMAL | PEAK | AVERAGE | HRESOLUTION) |
| :WAVeform:UNSigned {{0 | OFF} | {1 | ON}} (see page 461) | :WAVeform:UNSigned? (see page 461) | (0 | 1) |
| :WAVeform:VIEW <view> (see page 462) | :WAVeform:VIEW? (see page 462) | <view> ::= {MAIN} |
| n/a     | :WAVeform:XINCREMENT? (see page 463) | <return_value> ::= x-increment in the current preamble in NR3 format |
| n/a     | :WAVeform:XORIGIN? (see page 464) | <return_value> ::= x-origin value in the current preamble in NR3 format |
| n/a     | :WAVeform:XREFERENCE? (see page 465) | <return_value> ::= 0 (x-reference value in the current preamble in NR1 format) |
| n/a     | :WAVeform:YINCREMENT? (see page 466) | <return_value> ::= y-increment value in the current preamble in NR3 format |
### Table 32  :WAVeform Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:WAVeform:YORigin? (see page 467)</td>
<td>&lt;return_value&gt; ::= y-origin in the current preamble in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:YREFerence? (see page 468)</td>
<td>&lt;return_value&gt; ::= y-reference value in the current preamble in NR1 format</td>
</tr>
</tbody>
</table>
Syntax Elements

- "Number Format" on page 70
- "<NL> (Line Terminator)" on page 70
- "[ ] (Optional Syntax Terms)" on page 70
- "{ } (Braces)" on page 70
- "::= (Defined As)" on page 70
- "< > (Angle Brackets)" on page 71
- "... (Ellipsis)" on page 71
- "n,..,p (Value Ranges)" on page 71
- "d (Digits)" on page 71
- "Quoted ASCII String" on page 71
- "Definite-Length Block Response Data" on page 71

Number Format

NR1 specifies integer data.

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

<NL> (Line Terminator)

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

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

[ ] (Optional Syntax Terms)

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

{ } (Braces)

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

::= (Defined As)

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

$<>$ (Angle Brackets)

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

$...$ (Ellipsis)

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

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

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

d (Digits)

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

Quoted ASCII String

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

```vbnet
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
### Commands by Subsystem

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Common (*) Commands&quot; on page 75</td>
<td>Commands defined by IEEE 488.2 standard that are common to all instruments.</td>
</tr>
<tr>
<td>&quot;Root (: ) Commands&quot; on page 100</td>
<td>Control many of the basic functions of the oscilloscope and reside at the root level of the command tree.</td>
</tr>
<tr>
<td>&quot;:ACQuire Commands&quot; on page 138</td>
<td>Set the parameters for acquiring and storing data.</td>
</tr>
<tr>
<td>&quot;:BUS&lt;n&gt; Commands&quot; on page 150</td>
<td>Control all oscilloscope functions associated with the digital channels bus display.</td>
</tr>
<tr>
<td>&quot;:CALibrate Commands&quot; on page 159</td>
<td>Utility commands for determining the state of the calibration factor protection switch.</td>
</tr>
<tr>
<td>&quot;:CHANnel&lt;n&gt; Commands&quot; on page 167</td>
<td>Control all oscilloscope functions associated with individual analog channels or groups of channels.</td>
</tr>
<tr>
<td>&quot;:DIGital&lt;n&gt; Commands&quot; on page 186</td>
<td>Control all oscilloscope functions associated with individual digital channels.</td>
</tr>
<tr>
<td>&quot;:DISPlay Commands&quot; on page 193</td>
<td>Control how waveforms, graticule, and text are displayed and written on the screen.</td>
</tr>
<tr>
<td>&quot;:EXTernal Trigger Commands&quot; on page 203</td>
<td>Control the input characteristics of the external trigger input.</td>
</tr>
<tr>
<td>&quot;:FUNCTION Commands&quot; on page 212</td>
<td>Control functions in the measurement/storage module.</td>
</tr>
<tr>
<td>&quot;:HARDcopy Commands&quot; on page 224</td>
<td>Set and query the selection of hardcopy device and formatting options.</td>
</tr>
<tr>
<td>&quot;:MARKer Commands&quot; on page 232</td>
<td>Set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors).</td>
</tr>
<tr>
<td>&quot;:MEASURE Commands&quot; on page 243</td>
<td>Select automatic measurements to be made and control time markers.</td>
</tr>
</tbody>
</table>
### Command Types

Three types of commands are used:

- **Common (*) Commands** — See "Introduction to Common (*) Commands" on page 77 for more information.

- **Root Level (:) Commands** — See "Introduction to Root (:) Commands" on page 102 for more information.

- **Subsystem Commands** — Subsystem commands are grouped together under a common node of the "Command Tree" on page 569, such as the :TIMebase commands. Only one subsystem may be selected at any given time. When the instrument is initially turned on, the command parser is set to the root of the command tree; therefore, no subsystem is selected.
Common (*) Commands

Commands defined by IEEE 488.2 standard that are common to all instruments. See "Introduction to Common (*) Commands" on page 77.

Table 33  Common (*) Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS (see page 79)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>*ESE &lt;mask&gt; (see page 80)</td>
<td>*ESE? (see page 81)</td>
<td>&lt;mask&gt; ::= 0 to 255; an integer in NR1 format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit Weight Name Enables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- ------ ---- -----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 128 PON Power On</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6  64 URQ User Request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5  32 CME Command Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4   16 EXE Execution Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3   8 DDE Dev. Dependent Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2   4 QVE Query Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1   2 RQL Request Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0   1 OPC Operation Complete</td>
</tr>
<tr>
<td>n/a</td>
<td>*ESR? (see page 82)</td>
<td>&lt;status&gt; ::= 0 to 255; an integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>*IDN? (see page 82)</td>
<td>AGILENT TECHNOLOGIES,&lt;model&gt;,&lt;serial number&gt;,X.XX.XX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;model&gt; ::= the model number of the instrument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;serial number&gt; ::= the serial number of the instrument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;X.XX.XX&gt; ::= the software revision of the instrument</td>
</tr>
<tr>
<td>n/a</td>
<td>*LRN? (see page 85)</td>
<td>&lt;learn_string&gt; ::= current instrument setup as a block of data in IEEE 488.2 # format</td>
</tr>
<tr>
<td>*OPC (see page 86)</td>
<td>*OPC? (see page 86)</td>
<td>ASCII &quot;1&quot; is placed in the output queue when all pending device operations have completed.</td>
</tr>
</tbody>
</table>
### Table 33  Common (*) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>*OPT? (see page 87)</td>
<td>&lt;return_value&gt; ::= 0,0,&lt;license info&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;license info&gt; ::= &lt;All field&gt;, &lt;reserved&gt;, &lt;Factory MSO&gt;, &lt;Upgraded MSO&gt;, &lt;Probe field&gt;, &lt;Memory&gt;, &lt;Low Speed Serial&gt;, &lt;reserved&gt;, &lt;reserved&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;All field&gt; ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;reserved&gt; ::= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Factory MSO&gt; ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Upgraded MSO&gt; ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Probe field&gt; ::= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Memory&gt; ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Low Speed Serial&gt; ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;reserved&gt; ::= 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;reserved&gt; ::= 0</td>
</tr>
</tbody>
</table>

| *RCL <value> (see page 88) | n/a | <value> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) |

| *RST (see page 89) | n/a | See *RST (Reset) (see page 89) |

| *SAV <value> (see page 92) | n/a | <value> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9) |

<p>| *SRE &lt;mask&gt; (see page 93) | *SRE? (see page 94) | &lt;mask&gt; ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. &lt;mask&gt; ::= following values: |</p>
<table>
<thead>
<tr>
<th>Bit Weight</th>
<th>Name</th>
<th>Enables</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>OPER Operation Status Reg</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ESB Event Status Bit</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MAV Message Available</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>USR User</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TRG Trigger</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MSG Message</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

### Table 33  Common (*) Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>*STB? (see page 95)</td>
<td>&lt;value&gt; ::= 0 to 255; an integer in NR1 format, as shown in the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit Weight Name 1* Indicates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--- ------ ---- ---------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7  128 OPER Operation status condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6  64  RQS/Instrument is MSS requesting service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5  32  ESB Enabled event status condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4  16  MAV Message available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3  8    (Not used.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2  4    MSG Message displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1  2    USR User event condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  1    TRG A trigger occurred.</td>
</tr>
<tr>
<td>n/a</td>
<td>*TST? (see page 98)</td>
<td>&lt;result&gt; ::= 0 or non-zero value; an integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>*WAI (see page 99)</td>
<td>n/a</td>
</tr>
</tbody>
</table>
NOTE Each of the status registers has an enable (mask) register. By setting the bits in the enable register, you can select the status information you want to use.
**CLS (Clear Status)**

(see page 564)

**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 77
- "**STB (Read Status Byte)**" on page 95
- "**ESE (Standard Event Status Enable)**" on page 80
- "**ESR (Standard Event Status Register)**" on page 82
- "**SRE (Service Request Enable)**" on page 93
- ":SYSTem:ERRor" on page 315
3 Commands by Subsystem

**ESE (Standard Event Status Enable)**

(see page 564)

**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.

![Diagram showing Standard Event Status Enable (ESE) and Standard Event Status Register (ESR)](image)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PON</td>
<td>Power On</td>
<td>Event when an OFF to ON transition occurs.</td>
</tr>
<tr>
<td>6</td>
<td>URQ</td>
<td>User Request</td>
<td>Event when a front-panel key is pressed.</td>
</tr>
<tr>
<td>5</td>
<td>CME</td>
<td>Command Error</td>
<td>Event when a command error is detected.</td>
</tr>
<tr>
<td>4</td>
<td>EXE</td>
<td>Execution Error</td>
<td>Event when an execution error is detected.</td>
</tr>
<tr>
<td>3</td>
<td>DDE</td>
<td>Device Dependent Error</td>
<td>Event when a device-dependent error is detected.</td>
</tr>
<tr>
<td>2</td>
<td>QYE</td>
<td>Query Error</td>
<td>Event when a query error is detected.</td>
</tr>
</tbody>
</table>

*Table 34 Standard Event Status Enable (ESE)*
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 77
- "*ESR (Standard Event Status Register)" on page 82
- "*OPC (Operation Complete)" on page 86
- "*CLS (Clear Status)" on page 79

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RQL</td>
<td>Request Control</td>
<td>Event when the device is requesting control. (Not used.)</td>
</tr>
<tr>
<td>0</td>
<td>OPC</td>
<td>Operation Complete</td>
<td>Event when an operation is complete.</td>
</tr>
</tbody>
</table>
*ESR (Standard Event Status Register)

(see page 564)

Query Syntax

*ESR?

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PON</td>
<td>Power On</td>
<td>An OFF to ON transition has occurred.</td>
</tr>
<tr>
<td>6</td>
<td>URQ</td>
<td>User Request</td>
<td>A front-panel key has been pressed.</td>
</tr>
<tr>
<td>5</td>
<td>CME</td>
<td>Command Error</td>
<td>A command error has been detected.</td>
</tr>
<tr>
<td>4</td>
<td>EXE</td>
<td>Execution Error</td>
<td>An execution error has been detected.</td>
</tr>
<tr>
<td>3</td>
<td>DDE</td>
<td>Device Dependent Error</td>
<td>A device-dependent error has been detected.</td>
</tr>
<tr>
<td>2</td>
<td>QYE</td>
<td>Query Error</td>
<td>A query error has been detected.</td>
</tr>
</tbody>
</table>
Table 35  Standard Event Status Register (ESR) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RQL</td>
<td>Request Control</td>
<td>The device is requesting control. (Not used.)</td>
</tr>
<tr>
<td>0</td>
<td>OPC</td>
<td>Operation Complete</td>
<td>Operation is complete.</td>
</tr>
</tbody>
</table>

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 77
- "*ESE (Standard Event Status Enable)" on page 80
- "*OPC (Operation Complete)" on page 86
- "*CLS (Clear Status)" on page 79
- ":SYSTem:ERRor" on page 315
*IDN (Identification Number)

(see page 564)

Query Syntax

*IDN?

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

Return Format

AGILENT TECHNOLOGIES,<model>,<serial number>,X.XX.XX <NL>

<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 77
- "*OPT (Option Identification)" on page 87
**LRN (Learn Device Setup)**

(see page 564)

**Query Syntax**

*LRN?

The *LRN? query result contains the current state of the instrument. This query is similar to the :SYSTem:SETup? (see page 317) 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 Agilent 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 77
- "*RCL (Recall)" on page 88
- "*SAV (Save)" on page 92
- "::SYSTem:SETup" on page 317
**Commands by Subsystem**

*OPC (Operation Complete)*

(see page 564)

**Command Syntax**

*OPC

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

**Query Syntax**

*OPC?

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

**Return Format**

<complete><NL>

<complete> ::= 1

**See Also**

- "Introduction to Common (*) Commands" on page 77
- "*ESE (Standard Event Status Enable)" on page 80
- "*ESR (Standard Event Status Register)" on page 82
- "*CLS (Clear Status)" on page 79
**OPT (Option Identification)**

(see page 564)

**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>,<Factory MSO>,<Upgraded MSO>,
        <Probe field>,<Memory>,<Low Speed Serial>,<reserved>,
        <reserved>

<All field> ::= (0 | All)
<reserved> ::= 0
<Factory MSO> ::= (0 | MSO)
<Upgraded MSO> ::= (0 | MSO)
<Probe field> ::= 0
<Memory> ::= (0 | mem2M | mem8M)
<Low Speed Serial> ::= (0 | LSS)
<reserved> ::= 0
<reserved> ::= 0

The <Factory MSO> <Upgraded MSO> fields indicate whether the unit is a mixed-signal oscilloscope and, if so, whether it was factory installed or upgraded from an analog channels only oscilloscope (DSO).

The *OPT? query returns the following:

<table>
<thead>
<tr>
<th>Module</th>
<th>Module Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>No modules attached</td>
<td>0,0,0,0,MSO,0,0,mem8M,0,0,0</td>
</tr>
</tbody>
</table>

**See Also**

- "Introduction to Common (*) Commands" on page 77
- "*IDN (Identification Number)" on page 84
3 Commands by Subsystem

*RCL (Recall)
C
(see page 564)

Command Syntax
*RCL <value>

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

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

See Also
- "Introduction to Common (*) Commands" on page 77
- "*SAV (Save)" on page 92
**RST (Reset)**

(see page 564)

**Command Syntax**

The *RST command places the instrument in a known state. Reset conditions are:

<table>
<thead>
<tr>
<th>Acquire Menu</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Normal</td>
</tr>
<tr>
<td>Realtime</td>
<td>On</td>
</tr>
<tr>
<td>Averaging</td>
<td>Off</td>
</tr>
<tr>
<td># Averages</td>
<td>8</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Cursor Menu</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Channel 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital Channel Menu (MSO models only)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 0 - 15</td>
<td>Off</td>
</tr>
</tbody>
</table>
### Digital Channel Menu (MSO models only)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels</td>
<td>Off</td>
</tr>
<tr>
<td>Threshold</td>
<td>TTL (1.4V)</td>
</tr>
</tbody>
</table>

### Display Menu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definite persistence</td>
<td>Off</td>
</tr>
<tr>
<td>Grid</td>
<td>33%</td>
</tr>
<tr>
<td>Vectors</td>
<td>On</td>
</tr>
</tbody>
</table>

### Quick Meas Menu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Channel 1</td>
</tr>
</tbody>
</table>

### Run Control

Scope is running

### Time Base Menu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main time/division</td>
<td>100 us</td>
</tr>
<tr>
<td>Main time base delay</td>
<td>0.00 s</td>
</tr>
<tr>
<td>Delay time/division</td>
<td>500 ns</td>
</tr>
<tr>
<td>Delay time base delay</td>
<td>0.00 s</td>
</tr>
<tr>
<td>Reference</td>
<td>center</td>
</tr>
<tr>
<td>Mode</td>
<td>main</td>
</tr>
<tr>
<td>Vernier</td>
<td>Off</td>
</tr>
</tbody>
</table>

### Trigger Menu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Edge</td>
</tr>
<tr>
<td>Mode</td>
<td>Auto</td>
</tr>
<tr>
<td>Coupling</td>
<td>dc</td>
</tr>
<tr>
<td>Source</td>
<td>Channel 1</td>
</tr>
<tr>
<td>Level</td>
<td>0.0 V</td>
</tr>
<tr>
<td>Slope</td>
<td>Positive</td>
</tr>
<tr>
<td>HF Reject and noise reject</td>
<td>Off</td>
</tr>
</tbody>
</table>
### Trigger Menu

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Holdoff</td>
<td>60 ns</td>
</tr>
<tr>
<td>External probe attenuation</td>
<td>AutoProbe (if AutoProbe is connected), otherwise 1.0:1</td>
</tr>
<tr>
<td>External Units</td>
<td>Volts</td>
</tr>
<tr>
<td>External Impedance</td>
<td>1 M Ohm</td>
</tr>
</tbody>
</table>

#### See Also
- "Introduction to Common (*) Commands" on page 77

#### 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.
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**SAV (Save)**

(see page 564)

**Command Syntax**

*SAV <value>

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

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

**See Also**

- "Introduction to Common (\*) Commands" on page 77
- "*RCL (Recall)\*" on page 88
**SRE (Service Request Enable)**

(command syntax: [SRE <mask>]

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>OPER</td>
<td>Operation Status Register</td>
<td>Interrupts when enabled conditions in the Operation Status Register (OPER) occur.</td>
</tr>
<tr>
<td>6</td>
<td>...</td>
<td>...</td>
<td>(Not used.)</td>
</tr>
</tbody>
</table>
Table 36  Service Request Enable Register (SRE) (continued)

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

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 77
- "*STB (Read Status Byte)" on page 95
- "*CLS (Clear Status)" on page 79
**STB (Read Status Byte)**

(see page 564)

**Query Syntax**

*STB?*

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

**Return Format**

<value><NL>

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

---

**Table 37** Status Byte Register (STB)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>OPER</td>
<td>Operation Status Register</td>
<td>An enabled condition in the Operation Status Register (OPER) has occurred.</td>
</tr>
</tbody>
</table>
### Table 37  Status Byte Register (STB) (continued)

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

**NOTE**

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

**See Also**
- "Introduction to Common (*) Commands" on page 77
- "*SRE (Service Request Enable)" on page 93
**TRG (Trigger)**

(see page 564)

**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 77
- ":DIGitize" on page 111
- ":RUN" on page 131
- ":STOP" on page 135
**TST (Self Test)**  

(see page 564)

**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**  

\[
<\text{result}><NL>
\]

\[
<\text{result}> ::= 0 \text{ or non-zero value; an integer in NR1 format}
\]

**See Also**  

- "Introduction to Common (*) Commands" on page 77
*WAI (Wait To Continue)

(see page 564)

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 77
3 Commands by Subsystem

**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 102.

**Table 38 Root (:) Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:ACTivity (see page 103)</td>
<td>:ACTivity? (see page 103)</td>
<td>&lt;value&gt; ::= &lt;edges&gt;,&lt;levels&gt; (32-bit integer in NR1 format)</td>
</tr>
<tr>
<td><code>&lt;return value&gt;::= &lt;edges&gt;,&lt;levels&gt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:AER? (see page 104)</td>
<td>(0</td>
</tr>
<tr>
<td>:AUToscale [&lt;source&gt;[,...,&lt;source&gt;]] (see page 105)</td>
<td>n/a</td>
<td>&lt;source&gt;::= CHAnel&lt;n&gt; for DSO models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt;::= (CHAnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; can be repeated up to 5 times</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:AUToscale:AMODE &lt;value&gt; (see page 107)</td>
<td>:AUToscale:AMODE? (see page 107)</td>
<td>&lt;value&gt;::= {NORMal</td>
</tr>
<tr>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>:AUToscale:CHANnels &lt;value&gt; (see page 108)</td>
<td>:AUToscale:CHANnels? (see page 108)</td>
<td>&lt;value&gt;::= {ALL</td>
</tr>
<tr>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>:BLANK [&lt;source&gt;] (see page 109)</td>
<td>n/a</td>
<td>&lt;source&gt;::= (CHAnel&lt;n&gt;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt;::= (CHAnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CDISplay (see page 110)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| :DIGitize                     | n/a                        | <source> ::= (CHANnel<n> | FUNCTION | MATH | SBUS) for DSO models  
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15 | POD(1 | 2) | BUS(1 | 2) | FUNCTION | MATH | SBUS) for MSO models  
<source> can be repeated up to 5 times  
n ::= 1-2 or 1-4 in NR1 format |
| :HWEenable <n> (see page 113) | :HWEenable? (see page 113) | <n> ::= 16-bit integer in NR1 format                                                                                                                                 |
| :MERGe <pixel memory> (see page 119) | n/a | <pixel memory> ::= (PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9})  
| :OPEE <n> (see page 120) | :OPEE? (see page 121) | <n> ::= 16-bit integer in NR1 format                                                                                                                                 |
| :OVLENable <mask> (see page 126) | :OVLENable? (see page 127) | <mask> ::= 16-bit integer in NR1 format as shown:  
Bit Weight Input  
--- ------ -------  
10 1024 Ext Trigger Fault  
9 512 Channel 4 Fault  
8 256 Channel 3 Fault  
7 128 Channel 2 Fault  
6 64 Channel 1 Fault  
4 16 Ext Trigger OVL  
3 8 Channel 4 OVL  
2 4 Channel 3 OVL  
1 2 Channel 2 OVL  
0 1 Channel 1 OVL  
| n/a | :OVLRRegister? (see page 128) | <value> ::= integer in NR1 format.  
See OVLENable for <value> |
## 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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| ::PRINT [<options>] (see page 130) | n/a | `<options> ::= [<print option>,..,<print option>]`
| | | `<print option> ::= (COlor | GRAYscale | PRINter0 | BMP8bit | BMP | PNG | NOFactors | FACTors)`
| | | `<print option> can be repeated up to 5 times.` |
| ::RUN (see page 131) | n/a | n/a |
| n/a | ::SERial (see page 132) | `<return value> ::= unquoted string containing serial number` |
| n/a | ::SINGle (see page 133) | n/a | n/a |
| n/a | ::STATUS? <display> (see page 134) | `{0 | 1}` |
| | | `<display> ::= (CHANnel<n> | DIGital0,...,DIGital15 | POD{1 | 2} | BUS{1 | 2} | FUNCTION | MATH | SBUS)` |
| | | `<n> ::= 1-2 or 1-4 in NR1 format` |
| n/a | ::STOP (see page 135) | n/a | n/a |
| n/a | ::TER? (see page 136) | `{0 | 1}` |
| n/a | ::VIEW <source> (see page 137) | n/a | `<source> ::= (CHANnel<n> | PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | FUNCTION | MATH | SBUS}) for DSO models`
| | | `<source> ::= (CHANnel<n> | DIGital0,...,DIGital15 | PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | POD{1 | 2} | BUS{1 | 2} | FUNCTION | MATH | SBUS}) for MSO models`
| | | `<n> ::= 1-2 or 1-4 in NR1 format` |
:ACTivity
(see page 564)

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 102
- ":POD<n>:THReshold" on page 290
- ":DIGital<n>:THReshold" on page 192
:AER (Arm Event Register)

(see page 564)

Query Syntax

:AER?

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

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

Return Format

/value/<NL>

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

See Also

- "Introduction to Root (: Commands)" on page 102
- ":OPEE (Operation Status Enable Register)" on page 120
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 122
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 124
- "*STB (Read Status Byte)" on page 95
- "**SRE (Service Request Enable)" on page 93
**:AUToscale**

(see page 564)

**Command Syntax**

:AUToscale

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

<source> ::= CHANnel<n> for the DSO models

<source> ::= {DIGital0,..,DIGital15 | POD1 | POD2 | CHANnel<n>} for the MSO models

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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 Autoscale 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 108) 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.
- Trace memories.
- Delayed time base mode.

For further information on :AUToscale, see the User's Guide.
3 Commands by Subsystem

See Also

- "Introduction to Root (:) Commands" on page 102
- ":AUToscale:CHANnels" on page 108
- ":AUToscale:AMODE" on page 107

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 Autoscale key.

Example program from the start: "VISA COM Example in Visual Basic" on
page 614
:AUToscale:AMODE

(see page 564)

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 102
- ":AUToscale" on page 105
- ":AUToscale:CHANnels" on page 108
- ":ACQuire:TYPE" on page 148
- ":ACQuire:MODE" on page 144
:AUToscale:CHANnels

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 102
- ":AUToscale" on page 105
- ":AUToscale:AMODE" on page 107
- ":VIEW" on page 137
- ":BLANk" on page 109
:BLANk

(see page 564)

Command Syntax

:BLANk [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH | SBUS} for the DSO models
<source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | POD(1 | 2) | BUS(1 | 2) | FUNCtion | MATH | SBUS} for the MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models

The :BLANk command turns off (stops displaying) the specified channel, digital pod, math function, or serial decode bus. 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, or :DIGital<n>:DISPlay, are the preferred method to turn on/off a channel, etc.

NOTE

MATH is an alias for FUNCTION.

See Also

- "Introduction to Root (:) Commands" on page 102
- ":CDISplay" on page 110
- ":CHANnel<n>:DISPlay" on page 172
- ":DIGital<n>:DISPlay" on page 188
- ":FUNCTION:DISPlay" on page 215
- ":POD<n>:DISPlay" on page 288
- ":STATUs" on page 134
- ":VIEW" on page 137

Example Code

- "Example Code" on page 137
3 Commands by Subsystem

:CDISplay

(see page 564)

Command Syntax

:CDISplay

The :CDISplay 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 the data in active channels and functions is erased; however, new data is displayed on the next acquisition.

See Also

- "Introduction to Root (:) Commands" on page 102
- ":DISPlay:CLEar" on page 195
**:DIGitize**

(see page 564)

**Command Syntax**

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

- `<source>` ::= (CHANnel<n> | FUNCtion | MATH | SBUS) for the DSO models
- `<source>` ::= (CHANnel<n> | DIGital0,..,DIGital15 | POD1 | 2 | BUS1 | 2 | FUNCtion | MATH | SBUS) for the MSO models
- `<n>` ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
- `<n>` ::= (1 | 2) for the two channel oscilloscope models

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**

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

**NOTE**

MATH is an alias for FUNCtion.

**See Also**

- "Introduction to Root (:) Commands" on page 102
- ":RUN" on page 131
- ":SINGLE" on page 133
- ":STOP" on page 135
- ":ACQuire Commands" on page 138
- ":WAVEform Commands" on page 436

**Example Code**

```
'DIGITIZE - Used to acquire the waveform data for transfer over the interface. Sending this command causes an acquisition to take place with the resulting data being placed in the buffer.

' NOTE! The DIGITIZE command is highly recommended for triggering modes other than SINGLE. This ensures that sufficient data is available for measurement. If DIGITIZE is used with single mode, the completion criteria may never be met. The number of points gathered in Single mode is related to the sweep speed, memory depth, and maximum sample rate. For example, take an oscilloscope with a 1000-point memory, a sweep speed of 10 us/div (100 us total time across the screen), and a 20 MSa/s maximum sample rate. 1000 divided by 100 us equals 10 MSa/s. Because this number is
```
' less than or equal to the maximum sample rate, the full 1000 points
' will be digitized in a single acquisition. Now, use 1 us/div
' (10 us across the screen). 1000 divided by 10 us equals 100 MSa/s;
' because this is greater than the maximum sample rate by 5 times,
' only 400 points (or 1/5 the points) can be gathered on a single
' trigger. Keep in mind when the oscilloscope is running,
' communication with the computer interrupts data acquisition.
' Setting up the oscilloscope over the bus causes the data buffers
' to be cleared and internal hardware to be reconfigured. If a
' measurement is immediately requested, there may have not been
' enough time for the data acquisition process to collect data, and
' the results may not be accurate. An error value of 9.9E+37 may be
' returned over the bus in this situation.
'
myScope.WriteString "::DIGITIZE CHAN1"

Example program from the start: "VISA COM Example in Visual Basic" on
page 614
:HWEenable (Hardware Event Enable Register)

Command Syntax

:HWEnablen <mask>

<mask> ::= 16-bit integer

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

Query Syntax

:HWEnablen?

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

Return Format

<value><NL>

<value> ::= integer in NR1 format.

Table 39  Hardware Event Enable Register (HWEenable)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>PLL Locked</td>
<td>PLL Locked</td>
<td>This bit is for internal use and is not intended for general use.</td>
</tr>
<tr>
<td>11-1</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>0</td>
<td>Bat On</td>
<td>Battery On</td>
<td>Event when the battery is on.</td>
</tr>
</tbody>
</table>
See Also

- "Introduction to Root (:) Commands" on page 102
- ":AER (Arm Event Register)" on page 104
- ":CHANnel<n>:PROTect" on page 181
- ":EXTernal:PROTect" on page 209
- ":OPERRegister[:EVENt] (Operation Status Event Register)" on page 124
- ":OVLenable (Overload Event Enable Register)" on page 126
- ":OVLRegister (Overload Event Register)" on page 128
- "**STB (Read Status Byte)" on page 95
- "**SRE (Service Request Enable)" on page 93
:HWERegister:CONDition (Hardware Event Condition Register)

(see page 564)

Query Syntax

:HWERegister:CONDition?

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

Table 40 Hardware Event Condition Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>PLL Locked</td>
<td>PLL Locked</td>
<td>This bit is for internal use and is not intended for general use.</td>
</tr>
<tr>
<td>11-1</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>0</td>
<td>Bat On</td>
<td>Battery On</td>
<td>The battery is on.</td>
</tr>
</tbody>
</table>

Return Format

<value><NL>

<value> ::= integer in NR1 format.

See Also

- "Introduction to Root (:) Commands" on page 102
- ".CHANnel<n>:PROTection" on page 181
- ".EXTernal:PROTection" on page 209
- ".OPEE (Operation Status Enable Register)" on page 120
- ".OPERegister[:EVENT] (Operation Status Event Register)" on page 124
3 Commands by Subsystem

- "OVLenable (Overload Event Enable Register)" on page 126
- "OVLRegister (Overload Event Register)" on page 128
- "STB (Read Status Byte)" on page 95
- "SRE (Service Request Enable)" on page 93
:HWERegister[:EVENt] (Hardware Event Event Register)

(see page 564)

**Query Syntax**

:HWERegister[:EVENt]?

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

**Table 41** Hardware Event Event Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>PLL Locked</td>
<td>PLL Locked</td>
<td>This bit is for internal use and is not intended for general use.</td>
</tr>
<tr>
<td>11-1</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>0</td>
<td>Bat On</td>
<td>Battery On</td>
<td>The battery is on.</td>
</tr>
</tbody>
</table>

**Return Format**

<value><NL>

<value> ::= integer in NR1 format.

**See Also**

- "Introduction to Root (: ) Commands" on page 102
- "CHANnel<n>:PROTection" on page 181
- "EXTernal:PROTection" on page 209
- "OPEE (Operation Status Enable Register)" on page 120
- "OPERegister:CONDition (Operation Status Condition Register)" on page 122
3 Commands by Subsystem

- "OVLenable (Overload Event Enable Register)" on page 126
- "OVLRegister (Overload Event Register)" on page 128
- "STB (Read Status Byte)" on page 95
- "SRE (Service Request Enable)" on page 93
:MERGe

(see page 564)

Command Syntax

:MERGe <pixel memory>

<pixel memory> ::= {PMEMory0 | PMEMory1 | PMEMory2 | PMEMory3 |
                   PMEMory4 | PMEMory5 | PMEMory6 | PMEMory7 |
                   PMEMory8 | PMEMory9}

The :MERGe command stores the contents of the active display in the
specified pixel memory. The previous contents of the pixel memory are
overwritten. The pixel memories are PMEMory0 through PMEMory9. This
command is similar to the function of the "Save To: INTERN_<n>" key in
the Save/Recall menu.

See Also

- "Introduction to Root (:) Commands" on page 102
- "*SAV (Save)" on page 92
- "*RCL (Recall)" on page 88
- ":VIEW" on page 137
- ":BLANk" on page 109
:OPEE (Operation Status Enable Register)
(see page 564)

Command Syntax

`:OPEE <mask>`

<mask> ::= 16-bit integer

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

### Table 42 Operation Status Enable Register (OPEE)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>HWE</td>
<td>Hardware Event</td>
<td>Event when hardware event occurs.</td>
</tr>
<tr>
<td>11</td>
<td>OVLR</td>
<td>Overload</td>
<td>Event when 50Ω input overload occurs.</td>
</tr>
<tr>
<td>10-6</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>5</td>
<td>Wait Trig</td>
<td>Wait Trig</td>
<td>Event when the trigger is armed.</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
</tbody>
</table>
The :OPEE? query returns the current value contained in the Operation Status Enable register as an integer number.

Return Format

\[<\text{value}><\text{NL}>\]

\(<\text{value}> ::= \text{integer in NR1 format.}\]

See Also

- "Introduction to Root (: Commands)" on page 102
- ":AER (Arm Event Register)" on page 104
- ":CHANnel<n>:PROTection" on page 181
- ":EXTernal:PROTection" on page 209
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 124
- ":OVLenable (Overload Event Enable Register)" on page 126
- ":OVLRegister (Overload Event Register)" on page 128
- "*STB (Read Status Byte)" on page 95
- "*SRE (Service Request Enable)" on page 93
:OPERegister:CONDition (Operation Status Condition Register)

(see page 564)

Query Syntax :OPERegister:CONDition?

The :OPERegister:CONDition? query returns the integer value contained in the Operation Status Condition Register.

Table 43 Operation Status Condition Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>HWE</td>
<td>Hardware Event</td>
<td>A hardware event has occurred.</td>
</tr>
<tr>
<td>11</td>
<td>OVLR</td>
<td>Overload</td>
<td>A 50Ω input overload has occurred.</td>
</tr>
<tr>
<td>10-6</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>5</td>
<td>Wait Trig</td>
<td>Wait Trig</td>
<td>The trigger is armed (set by the Trigger Armed Event Register (TER)).</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>3</td>
<td>Run</td>
<td>Running</td>
<td>The oscilloscope is running (not stopped).</td>
</tr>
<tr>
<td>2-0</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
</tbody>
</table>
Return Format

\(<value><NL>\)

\(<value>: integer in NR1 format.\)

See Also

- "Introduction to Root (:) Commands" on page 102
- ":CHANnel<n>:PROTection" on page 181
- ":EXTernal:PROTection" on page 209
- ":OPEE (Operation Status Enable Register)" on page 120
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 124
- ":OVLEnable (Overload Event Enable Register)" on page 126
- ":OVLRegister (Overload Event Register)" on page 128
- "**STB (Read Status Byte)" on page 95
- "**SRE (Service Request Enable)" on page 93
- ":HWERRegister[:EVENt] (Hardware Event Event Register)" on page 117
- ":HWEenable (Hardware Event Enable Register)" on page 113
**:OPERegister[:EVENt] (Operation Status Event Register)**

(see page 564)

**Query Syntax**

`:OPERegister[:EVENt]?`

The :OPERegister[:EVENt]? query returns the integer value contained in the Operation Status Event Register.

---

**Table 44  Operation Status Event Register**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-13</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>12</td>
<td>HWE</td>
<td>Hardware Event</td>
<td>A hardware event has occurred.</td>
</tr>
<tr>
<td>11</td>
<td>OVLR</td>
<td>Overload</td>
<td>A 50Ω input overload has occurred.</td>
</tr>
<tr>
<td>10-6</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>5</td>
<td>Wait Trig</td>
<td>Wait Trig</td>
<td>The trigger is armed (set by the Trigger Armed Event Register (TER)).</td>
</tr>
<tr>
<td>4</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>3</td>
<td>Run</td>
<td>Running</td>
<td>The oscilloscope has gone from a stop state to a single or running state.</td>
</tr>
<tr>
<td>2-0</td>
<td>---</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
</tbody>
</table>
Return Format

<value><NL>

<value> ::= integer in NR1 format.

See Also

- "Introduction to Root (:) Commands" on page 102
- ":CHANnel<n>:PROTection" on page 181
- ":EXTernal:PROTection" on page 209
- ":OPEE (Operation Status Enable Register)" on page 120
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 122
- ":OVLenable (Overload Event Enable Register)" on page 126
- ":OVLRegister (Overload Event Register)" on page 128
- "*STB (Read Status Byte)" on page 95
- "*SRE (Service Request Enable)" on page 93
- ":HWERegister[:EVENt] (Hardware Event Event Register)" on page 117
- ":HWEenable (Hardware Event Enable Register)" on page 113
3 Commands by Subsystem

**:OVLenable (Overload Event Enable Register)**

(see page 564)

**Command Syntax**

`:OVLenable <enable_mask>`

`<enable_mask> ::= 16-bit integer`

The overload enable mask is an integer representing an input as described in the following table.

The :OVLenable command sets the mask in the Overload Event Enable Register and enables the reporting of the Overload Event Register. If an overvoltage is sensed on a 50Ω input, the input will automatically switch to 1 MΩ input impedance. If enabled, such an event will set bit 11 in the Operation Status Register.

**NOTE**

You can set analog channel input impedance to 50Ω on the 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models. On these same bandwidth models, if there are only two analog channels, you can also set external trigger input impedance to 50Ω.

### Table 45 Overload Event Enable Register (OVL)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-11</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>10</td>
<td>External Trigger Fault</td>
<td>Event when fault occurs on External Trigger input.</td>
</tr>
<tr>
<td>9</td>
<td>Channel 4 Fault</td>
<td>Event when fault occurs on Channel 4 input.</td>
</tr>
<tr>
<td>8</td>
<td>Channel 3 Fault</td>
<td>Event when fault occurs on Channel 3 input.</td>
</tr>
</tbody>
</table>
Table 45  Overload Event Enable Register (OVL) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>When Set (1 = High = True), Enables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Channel 2 Fault</td>
<td>Event when fault occurs on Channel 2 input.</td>
</tr>
<tr>
<td>6</td>
<td>Channel 1 Fault</td>
<td>Event when fault occurs on Channel 1 input.</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>4</td>
<td>External Trigger OVL</td>
<td>Event when overload occurs on External Trigger input.</td>
</tr>
<tr>
<td>3</td>
<td>Channel 4 OVL</td>
<td>Event when overload occurs on Channel 4 input.</td>
</tr>
<tr>
<td>2</td>
<td>Channel 3 OVL</td>
<td>Event when overload occurs on Channel 3 input.</td>
</tr>
<tr>
<td>1</td>
<td>Channel 2 OVL</td>
<td>Event when overload occurs on Channel 2 input.</td>
</tr>
<tr>
<td>0</td>
<td>Channel 1 OVL</td>
<td>Event when overload occurs on Channel 1 input.</td>
</tr>
</tbody>
</table>

Query Syntax

:OVLenable?

The :OVLenable query returns the current enable mask value contained in the Overload Event Enable Register.

Return Format

<enable_mask><NL>

<enable_mask> ::= integer in NR1 format.

See Also

- "Introduction to Root (:) Commands" on page 102
- ":CHANnel<n>:PROTection" on page 181
- ":EXTernal:PROTection" on page 209
- ":OPEE (Operation Status Enable Register)" on page 120
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 122
- ":OPERegister:[EVENT] (Operation Status Event Register)" on page 124
- ":OVLRegister (Overload Event Register)" on page 128
- "*STB (Read Status Byte)" on page 95
- "*SRE (Service Request Enable)" on page 93
### :OVLRegister (Overload Event Register)

**Query Syntax:**

`:OVLRegister?`

The :OVLRegister query returns the overload protection value stored in the Overload Event Register (OVLR). If an overvoltage is sensed on a 50Ω input, the input will automatically switch to 1 MΩ input impedance. A "1" indicates an overload has occurred.

**NOTE**

You can set analog channel input impedance to 50Ω on the 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models. On these same bandwidth models, if there are only two analog channels, you can also set external trigger input impedance to 50Ω.

---

**Table 46** Overload Event Register (OVLR)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-11</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
<tr>
<td>10</td>
<td>External Trigger Fault</td>
<td>Fault has occurred on External Trigger input.</td>
</tr>
<tr>
<td>9</td>
<td>Channel 4 Fault</td>
<td>Fault has occurred on Channel 4 input.</td>
</tr>
<tr>
<td>8</td>
<td>Channel 3 Fault</td>
<td>Fault has occurred on Channel 3 input.</td>
</tr>
<tr>
<td>7</td>
<td>Channel 2 Fault</td>
<td>Fault has occurred on Channel 2 input.</td>
</tr>
<tr>
<td>6</td>
<td>Channel 1 Fault</td>
<td>Fault has occurred on Channel 1 input.</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>(Not used.)</td>
</tr>
</tbody>
</table>
Table 46  Overload Event Register (OVLR) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>When Set (1 = High = True), Indicates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>External Trigger OVL</td>
<td>Overload has occurred on External Trigger input.</td>
</tr>
<tr>
<td>3</td>
<td>Channel 4 OVL</td>
<td>Overload has occurred on Channel 4 input.</td>
</tr>
<tr>
<td>2</td>
<td>Channel 3 OVL</td>
<td>Overload has occurred on Channel 3 input.</td>
</tr>
<tr>
<td>1</td>
<td>Channel 2 OVL</td>
<td>Overload has occurred on Channel 2 input.</td>
</tr>
<tr>
<td>0</td>
<td>Channel 1 OVL</td>
<td>Overload has occurred on Channel 1 input.</td>
</tr>
</tbody>
</table>

Return Format

\(<value><NL>\)

\(<value> ::= integer in NR1 format.\)

See Also

- "Introduction to Root (:) Commands" on page 102
- ":CHANnel<n>:PROTection" on page 181
- ":EXTernal:PROTection" on page 209
- ":OPEE (Operation Status Enable Register)" on page 120
- ":OVLenable (Overload Event Enable Register)" on page 126
- "*STB (Read Status Byte)" on page 95
- "*SRE (Service Request Enable)" on page 93
3 Commands by Subsystem

:PRINT

(see page 564)

Command Syntax

:PRINT [<options>]

<options> ::= [<print option>][,...,<print option>]

<print option> ::= (COLOR | GRAYscale | PRINTER0 | BMP8bit | BMP | PNG |
NOFactors | FACTors)

The <print option> parameter may be repeated up to 5 times.

The PRINT command formats the output according to the currently
selected format (device). If an option is not specified, the value selected in
the Print Config menu is used. Refer to "HARDcopy:FORMat" on page 228
for more information.

See Also

- "Introduction to Root (:) Commands" on page 102
- "Introduction to HARDcopy Commands" on page 224
- "HARDcopy:FORMat" on page 228
- "HARDcopy:FACTors" on page 225
- "HARDcopy:GRAYscale" on page 510
- "DISPLAY:DATA" on page 196
**:RUN**

(see page 564)

**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 102
- ":SINGle" on page 133
- ":STOP" on page 135

**Example Code**

```vbnet
' 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.
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614
3 Commands by Subsystem

:SERial

(see page 564)

Query Syntax: :SERial?

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

Return Format: Unquoted string

See Also: • "Introduction to Root (:) Commands" on page 102
Commands by Subsystem

:SINGle

(see page 564)

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 102
- ":RUN" on page 131
- ":STOP" on page 135
### :STATus

 niños (see page 564)

#### Query Syntax

:STATus? <source>

```plaintext
<source> ::= {CHANnel<n> | FUNCtion | MATH | SBUS} for the DSO models
<source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | POD{1 | 2} | BUS{1 | 2} | FUNCtion | MATH | SBUS} for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models
```

The :STATus? query reports whether the channel, function, trace memory, or serial decode bus specified by <source> is displayed.

#### Return Format

```plaintext
<value><NL>
<value> ::= {1 | 0}
```

#### See Also

- "Introduction to Root (:) Commands" on page 102
- ":BLANK" on page 109
- ":CHANnel<n>:DISPLAY" on page 172
- ":DIGital<n>:DISPLAY" on page 188
- ":FUNCTION:DISPLAY" on page 215
- ":POD<n>:DISPLAY" on page 288
- ":VIEW" on page 137

**NOTE**

MATH is an alias for FUNCTION.
:STOP

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 102
- "RUN" on page 131
- "SINGLE" on page 133

Example Code:
- "Example Code" on page 131
:TER (Trigger Event Register)

(see page 564)

Query Syntax

:TER?

The :TER? query reads the Trigger Event Register. After the Trigger Event Register is read, it is cleared. A one indicates a trigger has occurred. A zero indicates a trigger has not occurred.

The Trigger Event Register is summarized in the TRG bit of the Status Byte Register (STB). A Service Request (SRQ) can be generated when the TRG bit of the Status Byte transitions, and the TRG bit is set in the Service Request Enable register. The Trigger Event Register must be cleared each time you want a new service request to be generated.

Return Format

<value><NL>

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

See Also

- "Introduction to Root (: Commands" on page 102
- "*SRE (Service Request Enable)" on page 93
- "*STB (Read Status Byte)" on page 95
**:VIEW**

(see page 564)

**Command Syntax**

`:VIEW <source>`

`<source> ::= {CHANnel<n> | PMEMory0,...,PMEMory9 | FUNCtion | MATH | SBUS} for DSO models`

`<source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | PMEMory0,...,PMEMory9 | POD(1 | 2) | BUS(1 | 2) | FUNCtion | MATH | SBUS} for MSO models`

`<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models`

`<n> ::= (1 | 2) for the two channel oscilloscope models`

The :VIEW command turns on the specified channel, function, trace memory, or serial decode bus.

**NOTE**

MATH is an alias for FUNCTION.

**See Also**

- "Introduction to Root (:) Commands" on page 102
- "::BLANK" on page 109
- "::CHANnel<n>::DISPLAY" on page 172
- "::DIGital<n>::DISPLAY" on page 188
- "::FUNCtion::DISPLAY" on page 215
- "::POD<n>::DISPLAY" on page 288
- "::STATus" on page 134

**Example Code**

' VIEW_BLANK - (not executed in this example)'
' - VIEW turns on (starts displaying) a channel or pixel memory.'
' - BLANK turns off (stops displaying) a channel or pixel memory.'
' myScope.WriteString "::BLANK CHANNEL1"  ' Turn channel 1 off.'
' myScope.WriteString "::VIEW CHANNEL1"  ' Turn channel 1 on.'

Example program from the start: "VISA COM Example in Visual Basic" on page 614
3  Commands by Subsystem

:ACQuire Commands

Set the parameters for acquiring and storing data. See "Introduction to :ACQuire Commands" on page 138.

Table 47  :ACQuire Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:ACQuire:AALias? (see page 140)</td>
<td>(1</td>
</tr>
<tr>
<td>:ACQuire:COMPLETE &lt;complete&gt; (see page 141)</td>
<td>:ACQuire:COMPLETE? (see page 141)</td>
<td>&lt;complete&gt; ::= 100; an integer in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:COUNT &lt;count&gt; (see page 142)</td>
<td>:ACQuire:COUNT? (see page 142)</td>
<td>&lt;count&gt; ::= an integer from 1 to 65536 in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:DAALias &lt;mode&gt; (see page 143)</td>
<td>:ACQuire:DAALias? (see page 143)</td>
<td>&lt;mode&gt; ::= {DISable</td>
</tr>
<tr>
<td>:ACQuire:MODE &lt;mode&gt; (see page 144)</td>
<td>:ACQuire:MODE? (see page 144)</td>
<td>&lt;mode&gt; ::= {RTIMe</td>
</tr>
<tr>
<td>n/a</td>
<td>:ACQuire:POINts? (see page 145)</td>
<td>&lt;# points&gt; ::= an integer in NR1 format</td>
</tr>
<tr>
<td>:ACQuire:RSGnal &lt;ref_signal_mode&gt; (see page 146)</td>
<td>:ACQuire:RSGnal? (see page 146)</td>
<td>&lt;ref_signal_mode&gt; ::= {OFF</td>
</tr>
<tr>
<td>n/a</td>
<td>:ACQuire:SRATE? (see page 147)</td>
<td>&lt;sample_rate&gt; ::= sample rate (samples/s) in NR3 format</td>
</tr>
<tr>
<td>:ACQuire:TYPE &lt;type&gt; (see page 148)</td>
<td>:ACQuire:TYPE? (see page 148)</td>
<td>&lt;type&gt; ::= {NORMal</td>
</tr>
</tbody>
</table>

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. Two acquisition modes are available: real-time mode, and equivalent-time mode.

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 1 (smoothing) 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.

Real-time Mode

The :ACQuire:MODE RTIMe command sets the oscilloscope in real-time mode. This mode is useful to inhibit equivalent time sampling at fast sweep speeds.

Equivalent-time Mode

The :ACQuire:MODE ETIME command sets the oscilloscope in equivalent-time mode.

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
Commands by Subsystem

:ACQuire:AALias

(see page 564)

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 138
- ":ACQuire:DAALias" on page 143
**:ACQuire:COMPlete**

(see page 564)

**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 138
- ":ACQuire:TYPE" on page 148
- ":DIGitize" on page 111
- ":WAVEform:POINts" on page 449

**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"
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**:ACQuire:COUNt**

(see page 564)

**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 1 (smoothing) 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 1 to 65536 in NR1 format

**See Also**

- "Introduction to :ACQuire Commands" on page 138
- ":ACQuire:TYPE" on page 148
- ":DIGitize" on page 111
- ":WAVEform:COUNt" on page 445
**:ACQuire:**DAALias

(see page 564)

**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 138
- ":ACQuire:AALias" on page 140
**:ACQuire:MODE**

(see page 564)

**Command Syntax**

:ACQuire:MODE <mode>

<mode> ::= {RTIMe | ETIMe}

The :ACQuire:MODE command sets the acquisition mode of the oscilloscope. The :ACQuire:MODE RTIMe command sets the oscilloscope in real time mode. This mode is useful to inhibit equivalent time sampling at fast sweep speeds. The :ACQuire:MODE ETIME command sets the oscilloscope in equivalent time mode.

**NOTE**

The obsolete command ACQuire:TYPE:REALtime is functionally equivalent to sending ACQuire:MODE RTIMe; TYPE NORMal.

**Query Syntax**

:ACQuire:MODE?

The :ACQuire:MODE? query returns the acquisition mode of the oscilloscope.

**Return Format**

<mode_argument><NL>

<mode_argument> ::= {RTIM | ETIM}

**See Also**

- "Introduction to :ACQuire Commands" on page 138
- ":ACQuire:TYPE" on page 148
**:ACQuire:POINts**

(see page 564)

**Query Syntax**

**:ACQuire:POINts?**

The :ACQuire:POINts? query returns the number of data points that the hardware will acquire from the input signal. The number of points acquired is not directly controllable. To set the number of points to be transferred from the oscilloscope, use the command :WAVeform:POINts. The :WAVeform:POINts? query will return the number of points available to be transferred from the oscilloscope.

**Return Format**


<points_argument><NL>

<points_argument> ::= an integer in NR1 format

**See Also**

- "Introduction to :ACQuire Commands" on page 138
- " :DIGitize" on page 111
- " :WAVeform:POINts" on page 449
**:ACQuire:RSIGnal**

(see page 564)

**Command Syntax**

:ACQuire:RSIGnal <ref_signal_mode>

<ref_signal_mode> ::= {OFF | OUT | IN}

The :ACQuire:RSIGnal command selects the 10 MHz reference signal mode.

- The OFF mode disables the oscilloscope's 10 MHz REF BNC connector.
- The OUT mode is used to synchronize the timebase of two or more instruments.
- The IN mode is used to supply a sample clock to the oscilloscope. A 10 MHz square or sine wave signal is input to the BNC connector labeled 10 MHz REF. The amplitude must be between 180 mV and 1 V, with an offset of between 0 V and 2 V.

**CAUTION**

Do not apply more than ±15 V at the 10 MHz REF BNC connector on the rear panel, or damage to the instrument may occur.

**Query Syntax**

:ACQuire:RSIGnal?

The :ACQuire:RSIGnal? query returns the current 10 MHz reference signal mode.

**Return Format**

<ref_signal_mode><NL>

<ref_signal_mode> ::= {OFF | OUT | IN}

**See Also**

- ":TIMebase:REFClock" on page 325
- The *Agilent 6000 Series Oscilloscope User's Guide* for information on using the 10 MHz reference clock.
**:ACQuire:SRATe**

(see page 564)

**Query Syntax**

:ACQuire:SRATe?

The :ACQuire:SRATe? query returns the current oscilloscope acquisition sample rate. The sample rate is not directly controllable.

**Return Format**

`<sample_rate><NL>`

`<sample_rate> ::= sample rate in NR3 format`

**See Also**

- "Introduction to :ACQuire Commands" on page 138
- ":ACQuire:POINts" on page 145
**Command Syntax**

:ACQuire:TYPE <type>

<type> ::= {NORMal | AVERage | HRESolution | PEAK}

The :ACQuire:TYPE command selects the type of data acquisition that is to take place. The acquisition types are: NORMal, AVERage, HRESolution, and PEAK.

- The :ACQuire:TYPE NORMal command sets the oscilloscope in the normal mode.
- 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 1 (smoothing) to 65536. The COUNt value determines the number of averages that must be acquired.
- The :ACQuire:TYPE HRESolution command sets the oscilloscope in the high-resolution mode (also known as smoothing). This mode is used to reduce noise at slower sweep speeds where the digitizer samples faster than needed to fill memory for the displayed time range.

For example, if the digitizer samples at 200 MSa/s, but the effective sample rate is 1 MSa/s (because of a slower sweep speed), only 1 out of every 200 samples needs to be stored. Instead of storing one sample (and throwing others away), the 200 samples are averaged together to provide the value for one display point. The slower the sweep speed, the greater the number of samples that are averaged together for each display point.

This command is functionally equivalent to :ACQuire:TYPE AVERage and :ACQuire:COUNt 1.
- The :ACQuire:TYPE PEAK command sets the oscilloscope in the peak detect mode. In this mode, :ACQuire:COUNt has no meaning.

**NOTE**

The obsolete command ACQuire:TYPE:REALtime is functionally equivalent to sending ACQuire:MODE RTIME; TYPE NORMal.

**Query Syntax**

:ACQuire:TYPE?

The :ACQuire:TYPE? query returns the current acquisition type.

**Return Format**

<acq_type><NL>

<acq_type> ::= {NORM | AVER | HRES | PEAK}

**See Also**

- "Introduction to :ACQuire Commands" on page 138
- ":ACQuire:COUNt" on page 142
- ":ACQuire:MODE" on page 144
- ":DIGitize" on page 111
- ":WAVEform:TYPE" on page 460
- ":WAVEform:PREamble" on page 453

**Example Code**

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

Example program from the start: "VISA COM Example in Visual Basic" on page 614
 Commands by Subsystem

**:BUS<n> Commands**

Control all oscilloscope functions associated with buses made up of digital channels. See "Introduction to :BUS<n> Commands" on page 151.

### Table 48 :BUS<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:BUS&lt;n&gt;:BIT&lt;m&gt; {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:BITS &lt;channel_list&gt;, {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:CLEar (see page 155)</td>
<td>n/a</td>
<td>&lt;n&gt; ::= 1 or 2; an integer in NR1 format</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:DISPlay {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:LABel &lt;string&gt; (see page 157)</td>
<td>:BUS&lt;n&gt;:LABel? (see page 157)</td>
<td>&lt;string&gt; ::= quoted ASCII string up to 16 characters &lt;n&gt; ::= 1 or 2; an integer in NR1 format</td>
</tr>
<tr>
<td>:BUS&lt;n&gt;:MASK &lt;mask&gt; (see page 158)</td>
<td>:BUS&lt;n&gt;:MASK? (see page 158)</td>
<td>&lt;mask&gt; ::= 32-bit integer in decimal, &lt;nondecimal&gt;, or &lt;string&gt; &lt;nondecimal&gt; ::= #Hnn...n where n ::= {0,...,9</td>
</tr>
</tbody>
</table>
Introduction to :BUS<n> Commands

<n> ::= {1 | 2}

The BUS subsystem commands control the viewing, labeling, and digital channel makeup of two possible buses.

NOTE

These commands are only valid for the MSO models.

Reporting the Setup

Use :BUS<n>? to query setup information for the BUS subsystem.

Return Format

The following is a sample response from the :BUS1? query. In this case, the query was issued following a *RST command.

:BUS1:DISP 0;LAB "BUS1";MASK +255
**:BUS<n>:BIT<m>**

Command Syntax

```
:BUS<n>:BIT<m> <display>
```

- `<display>` ::= `{{1 | ON} | {0 | OFF}}`
- `<n>` ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.
- `<m>` ::= An integer, 0,...,15, is attached as a suffix to BIT and defines the digital channel that is affected by the command.

The :BUS<n>:BIT<m> command includes or excludes the selected bit as part of the definition for the selected bus. If the parameter is a 1 (ON), the bit is included in the definition. If the parameter is a 0 (OFF), the bit is excluded from the definition. Note: BIT0-15 correspond to DIGital0-15.

**NOTE**

This command is only valid for the MSO models.

Query Syntax

```
:BUS<n>:BIT<m>?
```

The :BUS<n>:BIT<m>? query returns the value indicating whether the specified bit is included or excluded from the specified bus definition.

Return Format

```
<display><NL>
```

- `<display>` ::= `{0 | 1}`

**See Also**

- "Introduction to :BUS<n> Commands" on page 151
- ":BUS<n>:BITS" on page 153
- ":BUS<n>:CLEar" on page 155
- ":BUS<n>:DISPLAY" on page 156
- ":BUS<n>:LABel" on page 157
- ":BUS<n>:MASK" on page 158

**Example Code**

```
' Include digital channel 1 in bus 1:
myScope.WriteString "::BUS1:BIT1 ON"
```
**:BUS\(<n>\):BITS**

(see page 564)

**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.

**NOTE**

This command is only valid for the MSO models.

**Query Syntax**

**:BUS\(<n>\):BITS?**

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

**Return Format**

**:BUS\(<n>\):BITS? <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 151
- ":BUS\(<n>\):BIT\(<m>\)" on page 152
- ":BUS\(<n>\):CLEar" on page 155
- ":BUS\(<n>\):DISPlay" on page 156
- ":BUS\(<n>\):LABel" on page 157
- ":BUS\(<n>\):MASK" on page 158

**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 (01:5,8,14), ON"
:BUS<n>:CLEar

Command Syntax

:BUS<n>:CLEar

<n> ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:CLEar command excludes all of the digital channels from the selected bus definition.

NOTE

This command is only valid for the MSO models.

See Also

- "Introduction to :BUS<n> Commands" on page 151
- ":BUS<n>:BIT<m>" on page 152
- ":BUS<n>:BITS" on page 153
- ":BUS<n>:DISPlay*" on page 156
- ":BUS<n>:LABel" on page 157
- ":BUS<n>:MASK" on page 158
3 Commands by Subsystem

:BUS<n>:DISPLAY

N (see page 564)

**Command Syntax**

:BUS<n>:DISPLAY <value>

<value> ::= {{1 | ON} | {0 | OFF}}

<n> ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:DISPLAY command enables or disables the view of the selected bus.

**NOTE**

This command is only valid for the MSO models.

**Query Syntax**

:BUS<n>:DISPLAY?

The :BUS<n>:DISPLAY? query returns the display value of the selected bus.

**Return Format**

<value><NL>

<value> ::= {0 | 1}

**See Also**

- "Introduction to :BUS<n> Commands" on page 151
- ":BUS<n>:BIT<m>" on page 152
- ":BUS<n>:BITS" on page 153
- ":BUS<n>:CLEar" on page 155
- ":BUS<n>:LABel" on page 157
- ":BUS<n>:MASK" on page 158
**:BUS<n>:LABel**

(see page 564)

**Command Syntax**

`:BUS<n>:LABel <quoted_string>`

- `<quoted_string>` ::= any series of 16 or less characters as a quoted ASCII string.
- `<n>` ::= An integer, 1 or 2, is attached as a suffix to BUS and defines the bus that is affected by the command.

The :BUS<n>:LABel command sets the bus label to the quoted string. Setting a label for a bus will also result in the name being added to the label list.

**NOTE**

This command is only valid for the MSO models.

**NOTE**

Label strings are 16 characters or less, and may contain any commonly used ASCII characters. Labels with more than 16 characters are truncated to 16 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 16 or less characters as a quoted ASCII string.

**See Also**

- "Introduction to :BUS<n> Commands" on page 151
- ":BUS<n>:BIT<m>" on page 152
- ":BUS<n>:BITS" on page 153
- ":BUS<n>:CLEar" on page 155
- ":BUS<n>:DISPLAY" on page 156
- ":BUS<n>:MASK" on page 158
- ":CHANnel<n>:LABel" on page 175
- ":DISPlay:LABList" on page 199
- ":DIGital<n>:LABel" on page 189

**Example Code**

' Set the bus 1 label to "Data":
myScope.WriteString "::BUS1:LABel 'Data'"
### :BUS<n>:MASK

(see page 564)

#### 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 151
- ":BUS<n>:BIT<m>" on page 152
- ":BUS<n>:BITS" on page 153
- ":BUS<n>:CLEar" on page 155
- ":BUS<n>:DISPlay" on page 156
- ":BUS<n>:LABel" on page 157

---

**NOTE**

This command is only valid for the MSO models.
Commands by Subsystem

:CALibrate Commands

Utility commands for viewing calibration status and for starting the user calibration procedure. See "Introduction to :CALibrate Commands" on page 159.

Table 49 :CALibrate Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:CALibrate:DATE? (see page 160)</td>
<td>&lt;return value&gt; ::= &lt;day&gt;,&lt;month&gt;,&lt;year&gt;; all in NR1 format</td>
</tr>
<tr>
<td>:CALibrate:LABel &lt;string&gt; (see page 161)</td>
<td>:CALibrate:LABel? (see page 161)</td>
<td>&lt;string&gt; ::= quoted ASCII string up to 32 characters</td>
</tr>
<tr>
<td>:CALibrate:STARt (see page 162)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:STATus? (see page 163)</td>
<td>&lt;return value&gt; ::= ALL,&lt;status_code&gt;,&lt;status_string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;status_code&gt; ::= an integer status code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;status_string&gt; ::= an ASCII status string</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:SWITch? (see page 164)</td>
<td>(PROTected</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:TEMPerature? (see page 165)</td>
<td>&lt;return value&gt; ::= degrees C delta since last cal in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:CALibrate:TIME? (see page 166)</td>
<td>&lt;return value&gt; ::= &lt;hours&gt;,&lt;minutes&gt;,&lt;seconds&gt;; all in NR1 format</td>
</tr>
</tbody>
</table>

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

(see page 564)

**Query Syntax**

```
:CALibrate:DATE?
```

The :CALibrate:DATE? query returns the date of the last calibration.

**Return Format**

```
<date><NL>
<date> ::= day,month,year in NR1 format<NL>
```

**See Also**

- "Introduction to :CALibrate Commands" on page 159
**:CALibrate:LABel**

(see page 564)

**Command Syntax**

`:CALibrate:LABel <string>`

<string> ::= quoted ASCII string of up to 32 characters in length, not including the quotes

The CALibrate:LABel command saves a string that is up to 32 characters in length into the instrument's non-volatile memory. The string may be used to record calibration dates or other information as needed.

**Query Syntax**

`:CALibrate:LABel?`

The :CALibrate:LABel? query returns the contents of the calibration label string.

**Return Format**

<string><NL>

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

**See Also**

- "Introduction to :CALibrate Commands" on page 159
### :CALibrate:STARt

**Command Syntax**

::CALibrate::STARt

The CALibrate:STARt command starts the user calibration procedure.

**NOTE**

Before starting the user calibration procedure, you must set the rear panel CALIBRATION switch to UNPROTECTED, and you must connect BNC cables from the TRIG OUT connector to the analog channel inputs. See the *User’s Guide* for details.

**See Also**

- "Introduction to :CALibrate Commands" on page 159
- "::CALibrate::SWITch" on page 164
**:CALibrate:STATus**

(see page 564)

**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> ::= ALL,<status_code>,<status_string>

<status_code> ::= an integer status code

<status_string> ::= an ASCII status string

**See Also**

- "Introduction to :CALibrate Commands" on page 159
3 Commands by Subsystem

:CALibrate:SWITch

(see page 564)

Query Syntax

:CALibrate:SWITch?

The :CALibrate:SWITch? query returns the rear-panel calibration protect (CAL PROTECT) switch state. The value PROTected indicates calibration is disabled, and UNPRotected indicates calibration is enabled.

Return Format

<switch><NL>

<switch> ::= {PROT | UNPR}

See Also

• "Introduction to :CALibrate Commands" on page 159
**:CALibrate:TEMPerature**

(see page 564)

**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 159
3 Commands by Subsystem

:CALibrate:TIME

(see page 564)

Query Syntax  :CALibrate:TIME?

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

Return Format  <date><NL>

<date> ::= hour,minutes,seconds in NR1 format

See Also  • "Introduction to :CALibrate Commands" on page 159
Commands by Subsystem

:CHANnel<n> Commands

Control all oscilloscope functions associated with individual analog channels or groups of channels. See "Introduction to :CHANnel<n> Commands" on page 168.

Table 50 :CHANnel<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CHANnel&lt;n&gt;:BWLimit {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:COUPling &lt;coupling&gt; (see page 171)</td>
<td>:CHANnel&lt;n&gt;:COUPling? (see page 171)</td>
<td>&lt;coupling&gt; ::= (AC</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:DISPlay {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:IMPedance &lt;impedance&gt; (see page 173)</td>
<td>:CHANnel&lt;n&gt;:IMPedance? (see page 173)</td>
<td>&lt;impedance&gt; ::= (ONEMeg</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:INVert {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:LABel &lt;string&gt; (see page 175)</td>
<td>:CHANnel&lt;n&gt;:LABel? (see page 175)</td>
<td>&lt;string&gt; ::= any series of 6 or less ASCII characters enclosed in quotation marks</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:OFFSet &lt;offset&gt;[suffix] (see page 176)</td>
<td>:CHANnel&lt;n&gt;:OFFSet? (see page 176)</td>
<td>&lt;offset&gt; ::= Vertical offset value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= (V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4; in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe &lt;attenuation&gt; (see page 177)</td>
<td>:CHANnel&lt;n&gt;:PROBe? (see page 177)</td>
<td>&lt;attenuation&gt; ::= Probe attenuation ratio in NR3 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:ID? (see page 178)</td>
<td>:CHANnel&lt;n&gt;:PROBe:ID? (see page 178)</td>
<td>&lt;probe id&gt; ::= unquoted ASCII string up to 11 characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:SKEW &lt;skew_value&gt; (see page 179)</td>
<td>:CHANnel&lt;n&gt;:PROBe:SKEW? (see page 179)</td>
<td>&lt;skew_value&gt; ::= -100 ns to +100 ns in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>
### Introduction to :CHANnel<n> Commands

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 6 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

---

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CHANnel&lt;n&gt;:PROBe:STYPe &lt;signal type&gt;</td>
<td>:CHANnel&lt;n&gt;:PROBe:STYPe?</td>
<td>&lt;signal type&gt; ::= (DIFFerential</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PROTection</td>
<td>:CHANnel&lt;n&gt;:PROTection?</td>
<td>(NORM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:RANGE &lt;range&gt;[suffix]</td>
<td>:CHANnel&lt;n&gt;:RANGE?</td>
<td>&lt;range&gt; ::= Vertical full-scale range value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= (V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:SCALe &lt;scale&gt;[suffix]</td>
<td>:CHANnel&lt;n&gt;:SCALe?</td>
<td>&lt;scale&gt; ::= Vertical units per division value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= (V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:UNITs &lt;units&gt;</td>
<td>:CHANnel&lt;n&gt;:UNITs?</td>
<td>&lt;units&gt; ::= (VOLTs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:VERNier {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>
Use \texttt{:CHANnel1?}, \texttt{:CHANnel2?}, \texttt{:CHANnel3?} or \texttt{:CHANnel4?} to query setup information for the CHANnel\textit{n} subsystem.

Return Format

The following are sample responses from the \texttt{:CHANnel\textit{n}?} query. In this case, the query was issued following a \texttt{*RST} command.

\begin{verbatim}
CHAN1:RANG +40.0E+00;OFFS +0.00000E+00;COUP DC;IMP ONEM;DISP 1;BWL 0;
INV 0;LAB "1";UNIT VOLT;PROB +1E+00;PROB:SKEW +0.00E+00;STYP SING
\end{verbatim}
:CHANnel<n>:BWLimit

(see page 564)

Command Syntax
:CHANnel<n>:BWLimit <bwlimit>

<brwlimit> ::= {{1 | ON} | {0 | OFF}}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :CHANnel<n>:BWLimit command controls an internal low-pass filter. When the filter is on, the bandwidth of the specified channel is limited to approximately 25 MHz.

Query Syntax
:CHANnel<n>:BWLimit?

The :CHANnel<n>:BWLimit? query returns the current setting of the low-pass filter.

Return Format
<bwlimit><NL>

<bwlimit> ::= {1 | 0}

See Also
• "Introduction to :CHANnel<n> Commands" on page 168
**:CHANnel<n>:COUPling**

(see page 564)

**Command Syntax**

`:CHANnel<n>:COUPling <coupling>

<coupling> ::= \{AC | DC\}

<n> ::= \{1 | 2 | 3 | 4\} for the four channel oscilloscope models

<n> ::= \{1 | 2\} for the two channel oscilloscope models

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 168
**:CHANnel<n>:DISPlay**

(see page 564)

**Command Syntax**

`:CHANnel<n>:DISPlay <display value>
<display value> ::= {{1 | ON} | {0 | OFF}}
<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models

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 168
- ":VIEW" on page 137
- ":BLANk" on page 109
- ":STATus" on page 134
- ":POD<n>:DISPlay" on page 288
- ":DIGital<n>:DISPlay" on page 188
**:CHANnel<n>:IMPedance**

(see page 564)

**Command Syntax**

:CHANnel<n>:IMPedance <impedance>

<impedance> ::= {ONEMeg | FIFTy}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :CHANnel<n>:IMPedance command selects the input impedance setting for the specified analog channel. The legal values for this command are ONEMeg (1 MΩ) and FIFTy (50Ω).

**NOTE**

The analog channel input impedance of the 100 MHz bandwidth oscilloscope models is fixed at ONEMeg (1 MΩ).

**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 168
3 Commands by Subsystem

:CHANnel\(<n>\):INVert

(see page 564)

**Command Syntax**

:CHANnel\(<n>\):INVert \(<invert\ value>\)

\(<invert\ value> ::= \{(1 | ON) | (0 | OFF)\}\)

\(<n> ::= (1 | 2 | 3 | 4)\) for the four channel oscilloscope models

\(<n> ::= (1 | 2)\) for the two channel oscilloscope models

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 168
\textbf{Command Syntax} \hspace{1cm} \texttt{:CHANnel<n>:LABel <string>}

\begin{itemize}
  \item <string> ::= quoted ASCII string
  \item <n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
  \item <n> ::= (1 | 2) for the two channel oscilloscope models
\end{itemize}

The \texttt{: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).

\textbf{Query Syntax} \hspace{1cm} \texttt{:CHANnel<n>:LABel?}

The \texttt{:CHANnel<n>:LABel?} query returns the label associated with a particular analog channel.

\textbf{Return Format} \hspace{1cm} <string><NL>

\begin{itemize}
  \item <string> ::= quoted ASCII string
\end{itemize}

\textbf{See Also} \hspace{1cm}
- "Introduction to \texttt{:CHANnel<n> Commands}" on page 168
- "\texttt{:DISPlay:LABel}" on page 198
- "\texttt{:DIGital<n>:LABel}" on page 189
- "\texttt{:DISPlay:LABList}" on page 199
- "\texttt{:BUS<n>:LABel}" on page 157

\textbf{Example Code} \hspace{1cm}
```vbnet
' LABEL - This command allows you to write a name (six characters maximum) next to the channel number. It is not necessary, but can be useful for organizing the display.
myScope.WriteString "':CHANNe1:LABEL ""CAL 1""" ' Label channel1 "CAL 1". myScope.WriteString "':CHANNe2:LABEL ""CAL2""" ' Label channel1 "CAL2".
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614

\textbf{NOTE} \hspace{1cm} Label strings are six characters or less, and may contain any commonly used ASCII characters. Labels with more than 6 characters are truncated to six characters. Lower case characters are converted to upper case.
### :CHANnel\(<n>\):OFFSet

(see page 564)

#### Command Syntax

\[:CHANnel\(<n>\):OFFSet \langle offset \rangle \ [\langle suffix \rangle]\]

- **\langle offset \rangle** := Vertical offset value in NR3 format
- **\langle suffix \rangle** := \(V \mid mV\)
- **\(<n>\)** := \{1 \mid 2 \mid 3 \mid 4\} for the four channel oscilloscope models
- **\(<n>\)** := \{1 \mid 2\} for the two channel oscilloscope models

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

\[\langle offset \rangle<\text{NL}>\]

- **\langle offset \rangle** := Vertical offset value in NR3 format

#### See Also

- "Introduction to :CHANnel\(<n>\) Commands" on page 168
- ":CHANnel\(<n>\):RANGe" on page 182
- ":CHANnel\(<n>\):SCALe" on page 183
- ":CHANnel\(<n>\):PROBe" on page 177
:CHANnel<n>:PROBe

(see page 564)

Command Syntax

:CHANnel<n>:PROBe <attenuation>

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

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The obsolete attenuation values X1, X10, X20, X100 are also supported.

The :CHANnel<n>:PROBe command specifies the probe attenuation factor for the selected channel. The probe attenuation factor may be 0.1 to 1000. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors, 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 168
- ":CHANnel<n>:RANGE" on page 182
- ":CHANnel<n>:SCALE" on page 183
- ":CHANnel<n>:OFFSET" on page 176

Example Code

' CHANNEL_PROBE - Sets the probe attenuation factor for the selected channel. The probe attenuation factor may be set from 0.1 to 1000.
myScope.WriteString ':CHAN1:PROBE 10' ' Set Probe to 10:1.

Example program from the start: "VISA COM Example in Visual Basic" on page 614
3 Commands by Subsystem

:CHANnel<n>:PROBe:ID

(see page 564)

Query Syntax

:CHANnel<n>:PROBe:ID?

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

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 168
:CHANnel<n>:PROBe:SKEW

(see page 564)

Command Syntax

:CHANnel<n>:PROBe:SKEW <skew value>

<skew value> ::= skew time in NR3 format
<skew value> ::= -100 ns to +100 ns
<n> ::= {1 | 2 | 3 | 4}

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 168
:CHANnel<n>:PROBe:STYPe

(see page 564)

Command Syntax

This command is valid only for the 113xA Series probes.

:CHANnel<n>:PROBe:STYPe <signal type>

<signal type> ::= (DIFFerential | SINGle)

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :CHANnel<n>:PROBe:STYPe command sets the channel probe signal type (STYPe) to differential or single-ended when using the 113xA Series probes and determines how offset is applied.

When single-ended is selected, the :CHANnel<n>:OFFset command changes the offset value of the probe amplifier. When differential is selected, the :CHANnel<n>:OFFset command changes the offset value of the channel amplifier.

Query Syntax

:CHANnel<n>:PROBe:STYPe?

The :CHANnel<n>:PROBe:STYPe? query returns the current probe signal type setting for the selected channel.

Return Format

<signal type><NL>

<signal type> ::= (DIFF | SING)

See Also

- "Introduction to :CHANnel<n> Commands" on page 168
- ":CHANnel<n>:OFFSet" on page 176
:CHANnel<n>:PROTection

\[^{\text{N}}\] (see page 564)

**Command Syntax**

:CHANnel<n>:PROTection[:CLEar]

\(<n>:\ {1|2|3|4}\)

When the analog channel input impedance is set to 50Ω (on the 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models), the input channels are protected against overvoltage. When an overvoltage condition is sensed, the input impedance for the channel is automatically changed to 1 MΩ. The :CHANnel<n>:PROTection[:CLEar] command is used to clear (reset) the overload protection. It allows the channel to be used again in 50Ω mode after the signal that caused the overload has been removed from the channel input. Reset the analog channel input impedance to 50Ω (see ":CHANnel<n>:IMPedance" on page 173) 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 168
- ":CHANnel<n>:COUPling" on page 171
- ":CHANnel<n>:IMPedance" on page 173
- ":CHANnel<n>:PROBe" on page 177
:CHANnel<n>:RANGe

(see page 564)

Command Syntax
:CHANnel<n>:RANGe <range>[<suffix>]

<range> ::= vertical full-scale range value in NR3 format
<suffix> ::= {V | mV}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :CHANnel<n>:RANGe command defines the full-scale vertical axis of the selected channel. When using 1:1 probe attenuation, the range can be set to any value from:

- 8 mV to 40 V for the 100 MHz models.
- 16 mV to 8 V for the 300 MHz – 1 GHz models with the input impedance set to 50Ω.

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 168
- ":CHANnel<n>:SCALe" on page 183
- ":CHANnel<n>:PROBe" on page 177

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.

Example program from the start: "VISA COM Example in Visual Basic" on page 614
:CHANnel<n>:SCALe

Command Syntax

:CHANnel<n>:SCALe <scale>[<suffix>]

<scale> ::= vertical units per division in NR3 format

<suffix> ::= \{V | mV\}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :CHANnel<n>:SCALe command sets the vertical scale, or units per division, of the selected channel. When using 1:1 probe attenuation, legal values for the scale range from:

- 1 mV to 5 V for the 100 MHz models.
- 2 mV to 1 V for the 300 MHz – 1 GHz models with the input impedance set to 50Ω.

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 168
- ":CHANnel<n>:RANGe" on page 182
- ":CHANnel<n>:PROBe" on page 177
Command Syntax

:CHANnel<n>:UNITs <units>

<units> ::= {VOLTs | AMPeres}

<n> ::= (1 | 2) for the two channel oscilloscope models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

The :CHANnel<n>:UNITs command sets the measurement units for the connected probe. Select VOLTs for a voltage probe and select AMPeres 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 168
- ":CHANnel<n>:RANGE" on page 182
- ":CHANnel<n>:PROBe" on page 177
- ":EXTernal:UNITs" on page 211
:CHANnel<n>:VERNier

Command Syntax
:CHANnel<n>:VERNier <vernier value>

<vornier value> ::= {{1 | ON} | {0 | OFF}}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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
<vornier value><NL>

<vornier value> ::= {0 | 1}

See Also
• "Introduction to :CHANnel<n> Commands" on page 168
3  Commands by Subsystem

:DIgital<n> Commands

Control all oscilloscope functions associated with individual digital channels. See "Introduction to :DIgital<n> Commands" on page 186.

Table 51 :DIgital<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DIgital&lt;n&gt;:DISPlay {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 0-15; an integer in NR1 format</td>
</tr>
<tr>
<td>:DIgital&lt;n&gt;:LABel &lt;string&gt; (see page 189)</td>
<td>:DIgital&lt;n&gt;:LABel? (see page 189)</td>
<td>&lt;string&gt; ::= any series of 6 or less ASCII characters enclosed in quotation marks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 0-15; an integer in NR1 format</td>
</tr>
<tr>
<td>:DIgital&lt;n&gt;:POSition &lt;position&gt; (see page 190)</td>
<td>:DIgital&lt;n&gt;:POSition? (see page 190)</td>
<td>&lt;position&gt; ::= 0-7 if display size = large, 0-15 if size = medium, 0-31 if size = small</td>
</tr>
<tr>
<td>:DIgital&lt;n&gt;:SIZE &lt;value&gt; (see page 191)</td>
<td>:DIgital&lt;n&gt;:SIZE? (see page 191)</td>
<td>&lt;value&gt; ::= {SMALl</td>
</tr>
<tr>
<td>:DIgital&lt;n&gt;:THReshold &lt;value&gt;[suffix] (see page 192)</td>
<td>:DIgital&lt;n&gt;:THReshold? (see page 192)</td>
<td>&lt;n&gt; ::= 0-15; an integer in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt; ::= {CMOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;user defined value&gt; ::= value in NR3 format from -8.00 to +8.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= {V</td>
</tr>
</tbody>
</table>

Introduction to :DIgital<n> Commands

<n> ::= {0,...,15}

The DIgital subsystem commands control the viewing, labeling, and positioning of digital channels. They also control threshold settings for groups of digital channels (D0-D7, D8-D15).

NOTE

These commands are only valid for the MSO models.

Reporting the Setup

Use :DIgital<n>? to query setup information for the DIgital subsystem.
Return Format

The following is a sample response from the :DIGital0? query. In this case, the query was issued following a *RST command.

:DIG0:DISP 0;THR +1.40E+00;LAB 'D0';POS +0
**:DIGital<n>:DISPlay**

(see page 564)

**Command Syntax**

:DIGital<n>:DISPlay <display>

<display> ::= {{1 | ON} | {0 | OFF}}

<n> ::= An integer, 0, 1,...15, is attached as a suffix to the command and defines the logic channel that is affected by the command.

The :DIGital<n>:DISPlay command turns digital display on or off for the specified channel.

**NOTE**

This command is only valid for the MSO models.

**Query Syntax**

:DIGital<n>:DISPlay?

The :DIGital<n>:DISPlay? query returns the current digital display setting for the specified channel.

**Return Format**

<display><NL>

<display> ::= {0 | 1}

**See Also**

- "Introduction to :DIGital<n> Commands" on page 186
- ":POD<n>:DISPlay" on page 288
- ":CHANnel<n>:DISPlay" on page 172
- ":VIEW" on page 137
- ":BLANk" on page 109
- ":STATus" on page 134
:DIGital<n>:LABel

Command Syntax

:DIGital<n>:LABel <string>

<string> ::= any series of 6 or less characters as quoted ASCII string.

<n> ::= An integer, 0,...,15, is attached as a suffix to the command and defines the logic channel that is affected by the command.

The :DIGital<n>:LABel command sets the channel label to the string that follows. Setting a label for a channel also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

NOTE

This command is only valid for the MSO models.

NOTE

Label strings are six characters or less, and may contain any commonly used ASCII characters. Labels with more than 6 characters are truncated to six characters.

Query Syntax

:DIGital<n>:LABel?

The :DIGital<n>:LABel? query returns the name of the specified channel.

Return Format

<string><NL>

<string> ::= any series of 6 or less characters as a quoted ASCII string.

See Also

- "Introduction to :DIGital<n> Commands" on page 186
- ":CHANnel<n>:LABel" on page 175
- ":DISPlay:LABList" on page 199
- ":BUS<n>:LABel" on page 157
**:DIGital<n>:POSition**

(see page 564)

**Command Syntax**

```
:DIGital<n>:POSition <position>
```

<position> ::= integer in NR1 format.

<n> ::= An integer, 0, 1,..,15, is attached as a suffix to the command and defines the logic channel that is affected by the command.

<table>
<thead>
<tr>
<th>Channel Size</th>
<th>Position</th>
<th>Top</th>
<th>Bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>0-7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Medium</td>
<td>0-15</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Small</td>
<td>0-31</td>
<td>31</td>
<td>0</td>
</tr>
</tbody>
</table>

The :DIGital<n>:POSition command sets the position of the specified channel.

**NOTE**

This command is only valid for the MSO models.

**Query Syntax**

```
:DIGital<n>:POSition?
```

The :DIGital<n>:POSition? query returns the position of the specified channel.

**Return Format**

```
<position><NL>
```

<position> ::= integer in NR1 format.

**See Also**

- "Introduction to :DIGital<n> Commands" on page 186
:DIGital<n>:SIZE

(see page 564)

**Command Syntax**

:DIGital<n>:SIZE <value>

<n> ::= An integer, 0, 1,...,15, is attached as a suffix to the command and defines the logic channel that is affected by the command.

<value> ::= {SMALL | MEDIUM | LARGE}

The :DIGital<n>: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 channels 1 through 15 as well.

**NOTE**

This command is only valid for the MSO models.

---

**Query Syntax**

:DIGital<n>:SIZE?

The :DIGital<n>:SIZE? query returns the size setting for the specified digital channels.

**Return Format**

<size_value><NL>

<size_value> ::= {SMALL | MEDIUM | LARGE}

**See Also**

- "Introduction to :DIGital<n> Commands" on page 186
- ":POD<n>:SIZE" on page 289
- ":DIGital<n>:POSITION" on page 190
### :DIGital<n>:THReshold

(see [page 564](#))

**Command Syntax**

`:DIGital<n>:THReshold <value>`

- `<value>` ::= (CMOS | ECL | TTL | <user defined value>[<suffix>])
- `<user defined value>` ::= -8.00 to +8.00 in NR3 format
- `<suffix>` ::= {V | mV | uV}
- `<n>` ::= An integer, 0, 1,...,15, is attached as a suffix to the command and defines the logic channel that is affected by the command.
  - TTL = 1.4V
  - CMOS = 2.5V
  - ECL = -1.3V

The :DIGital<n>:THReshold command sets the logic threshold value for all channels grouped with the specified channel (D0-D7, D8-D15). The threshold is used for triggering purposes and for displaying the digital data as high (above the threshold) or low (below the threshold).

#### NOTE

This command is only valid for the MSO models.

**Query Syntax**

`:DIGital<n>:THReshold?`

The :DIGital<n>: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<n> Commands" on page 186
- ":POD<n>:THReshold" on page 290
- ":TRIGger[:EDGE]:LEVel" on page 366
**:DISPlay Commands**

Control how waveforms, graticule, and text are displayed and written on the screen. See "Introduction to :DISPlay Commands" on page 193.

**Table 52 :DISPlay Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:DISPlay:CLEar (see page 195)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>:DISPlay:DATA [&lt;format&gt;][,&lt;][&lt;area&gt;][,&lt;][&lt;palette&gt;]&lt;display data&gt; (see page 196)</td>
<td>:DISPlay:DATA? [&lt;format&gt;][,&lt;][&lt;area&gt;][,&lt;][&lt;palette&gt;]&lt;display data&gt; (see page 196)</td>
<td>&lt;format&gt; ::= (TIFF) (command) &lt;area&gt; ::= (GRATicule) (command) &lt;palette&gt; ::= (MONochrome) (command) &lt;format&gt; ::= (TIFF</td>
</tr>
<tr>
<td>:DISPlay:LABel {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:DISPlay:LABList &lt;binary block&gt; (see page 199)</td>
<td>:DISPlay:LABList? (see page 199)</td>
<td>&lt;binary block&gt; ::= an ordered list of up to 75 labels, each 6 characters maximum, separated by newline characters</td>
</tr>
<tr>
<td>:DISPlay:PERSistence &lt;value&gt; (see page 200)</td>
<td>:DISPlay:PERSistence? (see page 200)</td>
<td>&lt;value&gt; ::= (MINimum</td>
</tr>
<tr>
<td>:DISPlay:SOURce &lt;value&gt; (see page 201)</td>
<td>:DISPlay:SOURce? (see page 201)</td>
<td>&lt;value&gt; ::= (PMEMory{0</td>
</tr>
<tr>
<td>:DISPlay:VECTors {{1</td>
<td>ON}</td>
<td>{0</td>
</tr>
</tbody>
</table>

**Introduction to :DISPlay Commands**

The DISPlay subsystem is used to control the display storage and retrieval of waveform data, labels, and text. This subsystem allows the following actions:

- Clear the waveform area on the display.
- Turn vectors on or off.
- Set waveform persistence.
• Specify labels.
• Save and Recall display data.

Reporting the Setup

Use :DISPlay? to query the setup information for the DISPlay subsystem.

Return Format

The following is a sample response from the :DISPlay? query. In this case, the query was issued following a *RST command.

:DISP:LAB 0;CONN 1;PERS MIN;SOUR PMEM9
**:DISPlay:CLEar**

(see page 564)

**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 193
- ":CDISplay" on page 110
3 Commands by Subsystem

:DISPlay:DATA

Command Syntax

:DISPlay:DATA [<format>][,][<area>][,][<palette>]<display data>

<format> ::= {TIFF}
<area> ::= {GRATicule}
<palette> ::= {MONochrome}
<display data> ::= binary block data in IEEE-488.2 # format.

The :DISPlay:DATA command writes trace memory data (a display bitmap) to the display or to one of the trace memories in the instrument.

If a data format or area is specified, the :DISPlay:DATA command transfers the data directly to the display. If neither the data format nor the area is specified, the command transfers data to the trace memory specified by the :DISPlay:SOURce command. Available trace memories are PMEMory0-9 and these memories correspond to the INTERN_0-9 files in the front panel Save/Recall menu.

Graticule data is a low resolution bitmap of the graticule area in TIFF format. This is the same data saved using the front panel Save/Recall menu or the *SAV (Save) command.

Query Syntax

:DISPlay:DATA? [<format>][,] [<area>][,] [<palette>]

<format> ::= (TIFF | BMP | BMP8bit | PNG)
<area> ::= (GRATicule | SCReen)
<palette> ::= (MONochrome | GRAYscale | COlor)

The :DISPlay:DATA? query reads display data from the screen or from one of the trace memories in the instrument. The format for the data transmission is the # format defined in the IEEE 488.2 specification.

If a data format or area is specified, the :DISPlay:DATA query transfers the data directly from the display. If neither the data format nor the area is specified, the query transfers data from the trace memory specified by the :DISPlay:SOURce command.

Screen data is the full display and is high resolution in grayscale or color. It may be read from the instrument in 24-bit bmp, 8-bit bmp, or 24-bit png format. This data cannot be sent back to the instrument.

Graticule data is a low resolution bitmap of the graticule area in TIFF format. You can get this data and send it back to the oscilloscope.
If the format is TIFF, the only valid value area parameter is GRATicule, and the only valid palette parameter is MONOchrome.

If the format is something other than TIFF, the only valid area parameter is SCReen, and the only valid values for palette are GRAYscale or COlor.

**Return Format**

<display data><NL>
<display data> ::= binary block data in IEEE-488.2 # format.

**See Also**

- "Introduction to :DISPlay Commands" on page 193
- ":DISPlay:SOURce" on page 201
- ":MERGe" on page 119
- ":PRINt" on page 130
- "*RCL (Recall)" on page 88
- "*SAV (Save)" on page 92
- "*VIEW" on page 137

**Example Code**

' IMAGE_TRANSFER - In this example, we will query for the image data ' with ":DISPLAY:DATA?", read the data, and then save it to a file.
Dim byteData() As Byte
myScope.IO.Timeout = 15000
myScope.WriteString ":DISPLAY:DATA? BMP, SCREEN, COLOR"
byteData = myScope.ReadIEEEBlock(BinaryType_UI1)
' Output display data to a file:
strPath = "c:\scope\data\screen.bmp"
' Remove file if it exists.
If Len(Dir(strPath)) Then
  Kill strPath
End If
Close #1 ' If #1 is open, close it.
Open strPath For Binary Access Write Lock Write As #1 ' Open file for output.
Put #1, , byteData ' Write data.
Close #1 ' Close file.
myScope.IO.Timeout = 5000

Example program from the start: "VISA COM Example in Visual Basic" on page 614
3 Commands by Subsystem

:DISPlay:LABel

(see page 564)

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 193
- "CHANnel<n>:LABel" on page 175

Example Code

' DISP_LABEL (not executed in this example)
  ' - Turns label names ON or OFF on the analyzer display.
  myScope.WriteString " :DISPLAY:LABEL ON" ' Turn on labels.

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**:DISPlay:LABList**

Command Syntax

**:DISPlay:LABList <binary block data>**

<binary block> ::= an ordered list of up to 75 labels, a maximum of six 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 "A12345", 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 six characters each, separated by newline characters.

See Also

- "Introduction to **:DISPlay Commands**" on page 193
- ":D:DISPlay:LABel" on page 198
- ":CHANnel<n>:LABel" on page 175
- ":DIGital<n>:LABel" on page 189
Commands by Subsystem

:DISPlay:PERSistence

(see page 564)

Command Syntax

:DISPlay:PERSistence <value>

<value> ::= (MINimum | INFinite)

The :DISPlay:PERSistence command specifies the persistence setting. MINimum indicates zero persistence and INFinite indicates infinite persistence. Use the :DISPlay:CLEar or :CDISplay root command to erase points stored by infinite persistence.

Query Syntax

:DISPlay:PERSistence?

The :DISPlay:PERSistence? query returns the specified persistence value.

Return Format

<value><NL>

<value> ::= (MIN | INF)

See Also

- "Introduction to :DISPlay Commands" on page 193
- ":DISPlay:CLEar" on page 195
- ":CDISplay" on page 110
### :DISPlay:SOURce

(see page 564)

#### Command Syntax

:DISPlay:SOURce <value>

<value> ::= {PMEM0 | PMEM1 | PMEM2 | PMEM3 | PMEM4
            | PMEM5 | PMEM6 | PMEM7 | PMEM8 | PMEM9}

PMEMory0-9 ::= pixel memory 0 through 9

The :DISPlay:SOURce command specifies the default source and destination for the :DISPlay:DATA command and query. PMEMory0-9 correspond to the INTERN_0-9 files found in the front panel Save/Recall menu.

#### Query Syntax

:DISPlay:SOURce?

The :DISPlay:SOURce? query returns the specified SOURce.

#### Return Format

<value><NL>

<value> ::= {PMEM0 | PMEM1 | PMEM2 | PMEM3 | PMEM4 | PMEM5 | PMEM6
            | PMEM7 | PMEM8 | PMEM9}

#### See Also

- "Introduction to :DISPlay Commands" on page 193
- ":DISPlay:DATA" on page 196
:DISPlay:VECTors

(see page 564)

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. When vectors are turned off, only the sampled data is displayed.

Query Syntax

:DISPlay:VECTors?

The :DISPlay:VECTors? query returns whether vector display is on or off.

Return Format

<vectors><NL>

<vectors> ::= {1 | 0}

See Also

• "Introduction to :DISPlay Commands" on page 193
**Commands by Subsystem**

---

**:EXTernal Trigger Commands**

Control the input characteristics of the external trigger input. See "Introduction to :EXTernal Trigger Commands" on page 203.

### Table 53 :EXTernal Trigger Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:EXTernal:BWLimit &lt;bwlimit&gt; (see page 204)</td>
<td>:EXTernal:BWLimit? (see page 204)</td>
<td>&lt;bwlimit&gt; ::= {0</td>
</tr>
<tr>
<td>:EXTernal:IMPedance &lt;value&gt; (see page 205)</td>
<td>:EXTernal:IMPedance? (see page 205)</td>
<td>&lt;impedance&gt; ::= {ONEMeg</td>
</tr>
<tr>
<td>:EXTernal:PROBe &lt;attenuation&gt; (see page 206)</td>
<td>:EXTernal:PROBe? (see page 206)</td>
<td>&lt;attenuation&gt; ::= probe attenuation ratio in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:EXTernal:PROBe:ID? (see page 207)</td>
<td>&lt;probe id&gt; ::= unquoted ASCII string up to 11 characters</td>
</tr>
<tr>
<td>:EXTernal:PROBe:STYPE &lt;signal type&gt; (see page 208)</td>
<td>:EXTernal:PROBe:STYPE? (see page 208)</td>
<td>&lt;signal type&gt; ::= {DIFFerential</td>
</tr>
<tr>
<td>:EXTernal:PROTection[:CLEar] (see page 209)</td>
<td>:EXTernal:PROTection? (see page 209)</td>
<td>{NORM</td>
</tr>
<tr>
<td>:EXTernal:RANGe &lt;range&gt;[&lt;suffix&gt;] (see page 210)</td>
<td>:EXTernal:RANGe? (see page 210)</td>
<td>&lt;range&gt; ::= vertical full-scale range value in NR3 format</td>
</tr>
<tr>
<td>:EXTernal:UNITs &lt;units&gt; (see page 211)</td>
<td>:EXTernal:UNITs? (see page 211)</td>
<td>&lt;units&gt; ::= {VOLTs</td>
</tr>
</tbody>
</table>

---

**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.

```plaintext
:EXT:BWLI 0; IMP ONEM; RANG +8.0E+00; UNIT VOLT; PROB +1.0E+00; PROB:STYPE SING
```
3  Commands by Subsystem

:EXTernal:BWLimit

(see page 564)

**Command Syntax**

```plaintext
: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**

```plaintext
:EXTernal:BWLimit?
```

The :EXTernal:BWLimit? query returns the current setting of the low-pass filter (always 0).

**Return Format**

```plaintext
<bwlimit><NL>
```

<bwlimit> ::= 0

**See Also**

- "Introduction to :EXTernal Trigger Commands" on page 203
- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:HFReject" on page 336
**:EXTernal:IMPedance**

(see page 564)

**Command Syntax**

:EXTernal:IMPedance <value>

<value> ::= {ONEMeg | FIFTy}

The :EXTernal:IMPedance command selects the input impedance setting for the external trigger. The legal values for this command are ONEMeg (1 MΩ) and FIFTy (50Ω).

**NOTE**

You can set external trigger input impedance to FIFTy (50Ω) on the 2-channel, 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models.

**Query Syntax**

:EXTernal:IMPedance?

The :EXTernal:IMPedance? query returns the current input impedance setting for the external trigger.

**Return Format**

<impedance value><NL>

<impedance value> ::= {ONEM | FIFT}

**See Also**

- "Introduction to :EXTernal Trigger Commands" on page 203
- "Introduction to :TRIgger Commands" on page 332
- ":CHANnel<n>:IMPedance" on page 173
:EXTernal:PROBe

(see page 564)

Command Syntax

:EXTernal:PROBe <attenuation>

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

The :EXTernal:PROBe command specifies the probe attenuation factor for the external trigger. The probe attenuation factor may be 0.1 to 1000. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors and for setting trigger levels.

If an AutoProbe probe is connected to the oscilloscope, the attenuation value cannot be changed from the sensed value. Attempting to set the oscilloscope to an attenuation value other than the sensed value produces an error.

Query Syntax

:EXTernal:PROBe?

The :EXTernal:PROBe? query returns the current probe attenuation factor for the external trigger.

Return Format

<attenuation><NL>

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

See Also

- "Introduction to :EXTernal Trigger Commands" on page 203
- ":EXTernal:RANGe" on page 210
- "Introduction to :TRIGger Commands" on page 332
- ":CHANnel<n>:PROBe" on page 177
:EXTernal:PROBe:ID

(see page 564)

Query Syntax :EXTernal:PROBe:ID?

The :EXTernal:PROBe:ID? query returns the type of probe attached to the external trigger input.

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 :EXTernal Trigger Commands" on page 203
Command Syntax

This command is valid only for the 113xA Series probes.

```
:EXTernal:PROBe:STYPe <signal type>
<signal type> ::= (DIFFerential | SINGle)
```

The :EXTernal:PROBe:STYPe command sets the external trigger probe signal type (STYPe) to differential or single-ended when using the 113xA Series probes and determines how offset is applied.

Query Syntax

```
:EXTernal:PROBe:STYPe?
```

The :EXTernal:PROBe:STYPe? query returns the current probe signal type setting for the external trigger.

Return Format

```
<signal type><NL>
<signal type> ::= (DIFF | SING)
```

See Also

- "Introduction to :EXTernal Trigger Commands" on page 203
**Command Syntax**

```
:EXTernal:PROTection[:CLEar]
```

When the external trigger input impedance is set to 50Ω (on the 2-channel, 300 MHz, 500 MHz, and 1 GHz bandwidth oscilloscope models), the external trigger input is protected against overvoltage. When an overvoltage condition is sensed, the input impedance for the external trigger is automatically changed to 1 MΩ. The :EXTernal:PROTection[:CLEar] command is used to clear (reset) the overload protection. It allows the external trigger to be used again in 50Ω mode after the signal that caused the overload has been removed from the external trigger input. Reset the external trigger input impedance to 50Ω (see ":EXTernal:IMPedance" on page 205) after clearing the overvoltage protection.

**Query Syntax**

```
:EXTernal:PROTection?
```

The :EXTernal:PROTection query returns the state of the input protection for external trigger. If the external trigger input has experienced an overload, TRIP (tripped) will be returned; otherwise NORM (normal) is returned.

**Return Format**

```
{NORM | TRIP}<NL>
```

**See Also**

- "Introduction to :EXTernal Trigger Commands" on page 203
- ":EXTernal:IMPedance" on page 205
- ":EXTernal:PROBe" on page 206
3 Commands by Subsystem

:EXTernal:RANGe

(see page 564)

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. The range can only be set to 5.0 V when using 1:1 probe attenuation. 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 = (5.0 V) * (probe attenuation factor)

See Also

- "Introduction to :EXTernal Trigger Commands" on page 203
- ":EXTernal:PROBe" on page 206
- "Introduction to :TRIGger Commands" on page 332
**:EXTernal:UNITs**

(see page 564)

**Command Syntax**

`:EXTernal:UNITs <units>`

<units> ::= {VOLTs | AMPeres}

The :EXTernal:UNITs command sets the measurement units for the probe connected to the external trigger input. Select VOLTs for a voltage probe and select AMPeres 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 203
- "Introduction to :TRIGger Commands" on page 332
- ":EXTernal:RANGe" on page 210
- ":EXTernal:PROBe" on page 206
- ":CHANnel<n>:UNITs" on page 184
## :FUNCTION Commands

Control functions in the measurement/storage module. See "Introduction to :FUNCTION Commands" on page 213.

### Table 54 :FUNCTION Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:FUNCTION:CENTer &lt;frequency&gt;</td>
<td>:FUNCTION:CENTer?</td>
<td>&lt;frequency&gt; ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 25 GHz.</td>
</tr>
<tr>
<td>(see page 214)</td>
<td>(see page 214)</td>
<td></td>
</tr>
<tr>
<td>:FUNCTION:DISPlay {{0</td>
<td>OFF}</td>
<td>:FUNCTION:DISPlay?</td>
</tr>
<tr>
<td>(1</td>
<td>ON}) (see page 215)</td>
<td>(see page 215)</td>
</tr>
<tr>
<td>:FUNCTION:OFFSet &lt;offset&gt;</td>
<td>:FUNCTION:OFFSet?</td>
<td>&lt;offset&gt; ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function.</td>
</tr>
<tr>
<td>(see page 216)</td>
<td>(see page 216)</td>
<td></td>
</tr>
<tr>
<td>:FUNCTION:OPERation &lt;operation&gt;</td>
<td>:FUNCTION:OPERation?</td>
<td>&lt;operation&gt; ::= {SUBTract</td>
</tr>
<tr>
<td>(see page 217)</td>
<td>(see page 217)</td>
<td></td>
</tr>
<tr>
<td>:FUNCTION:RANGe &lt;range&gt;</td>
<td>:FUNCTION:RANGe?</td>
<td>&lt;range&gt; ::= the full-scale vertical axis value in NR3 format.</td>
</tr>
<tr>
<td>(see page 218)</td>
<td>(see page 218)</td>
<td>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 DIFFerentiate function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV.</td>
</tr>
<tr>
<td>:FUNCTION:REFerence &lt;level&gt;</td>
<td>:FUNCTION:REFerence?</td>
<td>&lt;level&gt; ::= the current reference level in NR3 format.</td>
</tr>
<tr>
<td>(see page 219)</td>
<td>(see page 219)</td>
<td>The range of legal values is from 400.0 dBV to +400.0 dBV (depending on current range value).</td>
</tr>
<tr>
<td>:FUNCTION:SCALe &lt;scale value&gt;[&lt;suffix&gt;]</td>
<td>:FUNCTION:SCALe?</td>
<td>&lt;scale value&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>(see page 220)</td>
<td>(see page 220)</td>
<td>&lt;suffix&gt; ::= (V</td>
</tr>
</tbody>
</table>
Introduction to :FUNCTION Commands

The FUNCTION subsystem controls the math functions in the oscilloscope. Multiply (channel 1 x channel 2), subtract (channel 1 - channel 2), differentiate, integrate, and FFT (Fast Fourier Transform) operations are available. These math operations only use the analog (vertical) channels.

NOTE
To perform analog channel addition, set analog channel 2 to invert and select subtract (channel 1 - channel 2).

The SOURce, DISPlay, RANGE, and OFFSet commands apply to any function. The SPAN, CENTER, and WINDow commands are only useful for FFT functions. When FFT is selected, the cursors change from volts and time to decibels (dB) and frequency (Hz).

Reporting the Setup

Use :FUNCTION? to query setup information for the FUNCTION subsystem.

Return Format

The following is a sample response from the :FUNCTION? queries. In this case, the query was issued following a *RST command.

:FUNCTION:OPER SUBT;DISP 0;RANG +8.00E+00;OFFS +0.00000E+00
3 Charges by Subsystem

```
:FUNCtion:CENTer

(see page 564)

Command Syntax
:FUNCtion:CENTer <frequency>

<frequency> ::= the current center frequency in NR3 format. The range
of legal values is from 0 Hz to 25 GHz.

The :FUNCtion:CENTer command sets the center frequency when FFT
(Fast Fourier Transform) is selected.

Query Syntax
:FUNCtion:CENTer?

The :FUNCtion: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 0 Hz to 25 GHz.

NOTE
After a *RST (Reset) or :AUToscale command, the values returned by the
value. Once you change either the :FUNCtion:CENTer or :FUNCtion:SPAN value, they no
longer track the :TIMebase:RANGe value.

See Also
- "Introduction to :FUNCtion Commands" on page 213
- ":FUNCtion:SPAN" on page 222
- ":TIMebase:RANGe" on page 324
- ":TIMebase:SCALe" on page 327
```
Commands by Subsystem

:FUNCtion:DISPlay

(see page 564)

Command Syntax

:FUNCtion:DISPlay <display>

<display> ::= {{1 | ON} | {0 | OFF}}

The :FUNCtion:DISPlay command turns the display of the function on or off. When ON is selected, the function performs as specified using the other FUNCtion commands. When OFF is selected, function is neither calculated nor displayed.

Query Syntax

:FUNCtion:DISPlay?

The :FUNCtion:DISPlay? query returns whether the function display is on or off.

Return Format

<display><NL>

<display> ::= {1 | 0}

See Also

- "Introduction to :FUNCtion Commands" on page 213
- ":VIEW" on page 137
- ":BLANk" on page 109
- ":STATus" on page 134
### :FUNCtion:OFFSet

(see page 564)

**Command Syntax**

:FUNCtion:OFFSet <offset>

<offset> ::= the value at center screen in NR3 format.

The :FUNCtion: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:OFFset command is equivalent to the :FUNCtion:REReference command.

**Query Syntax**

:FUNCtion:OFFSet?

The :FUNCtion: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 Commands" on page 213
- ":FUNCtion:RANGE" on page 218
- ":FUNCtion:REReference" on page 219
- ":FUNCtion:SCALe" on page 220
:FUNCtion:OPERation

(see page 564)

Command Syntax

:FUNCtion:OPERation <operation>

<operation> ::= {SUBTract | MULTiply | INTegrate | DIFFerentiate | FFT | SQRT}

The :FUNCtion:OPERation command sets the desired operation for a function. (FFT = Fast Fourier Transform, SQRT = square root.)

Query Syntax

:FUNCtion:OPERation?

The :FUNCtion:OPERation? query returns the current operation for the selected function.

Return Format

<operation><NL>

<operation> ::= {SUBT | MULT | INT | DIFF | FFT | SQRT}

See Also

• "Introduction to :FUNCtion Commands" on page 213
**:FUNCtion:RANGe**

(see page 564)

**Command Syntax**

**:FUNCtion:RANGe <range>**

<range> ::= the full-scale vertical axis value in NR3 format.

The :FUNCtion:RANGe command defines the full-scale vertical axis for the selected function.

**Query Syntax**

**:FUNCtion:RANGe?**

The :FUNCtion: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.

The range for ADD, SUBT, MULT is 8E-6 to 800E+3.

The range for the INTEGRATE function is 8E-9 to 400E+3 (depends on sweep speed).

The range for the DIFFerentiate function is 80E-3 to 8.0E12 (depends on sweep speed).

The range for the FFT (Fast Fourier Transform) function is 8 to 800 dBV.

**See Also**

- "Introduction to :FUNCtion Commands" on page 213
- ":FUNCtion:SCALe" on page 220
"FUNCtion:REFerence"

(see page 564)

Command Syntax

:FUNCtion:REFerence <level>

<level> ::= the current reference level in NR3 format.

The range of legal values is from -400.0 dBV to +400.0 dBV depending on the current :FUNCtion:RANGe value. If you set the reference level to a value outside of the legal range, it is automatically set to the nearest legal value.

The :FUNCtion:REFerence command is only used when an FFT (Fast Fourier Transform) operation is selected. The :FUNCtion:REFerence command sets the reference level represented by center screen.

NOTE

The FUNCtion:REFerence command is equivalent to the :FUNCtion:OFFSet command.

Query Syntax

:FUNCtion:REFerence?

The :FUNCtion:REFerence? query returns the current reference level in dBV.

Return Format

<level><NL>

<level> ::= the current reference level in NR3 format.

See Also

- "Introduction to :FUNCtion Commands" on page 213
- ":FUNCtion:OFFSet" on page 216
- ":FUNCtion:RANGe" on page 218
- ":FUNCtion:SCALE" on page 220
**:FUNCtion:SCALe**

(see page 564)

**Command Syntax**

**:FUNCtion:SCALe** <scale value>[<suffix>]

<scale value> ::= integer in NR1 format

<suffix> ::= \( V \mid dB \)

The :**FUNCtion:SCALe** command sets the vertical scale, or units per division, of the selected function. Legal values for the scale depend on the selected function.

**Query Syntax**

**:FUNCtion:SCALe?**

The :**FUNCtion:SCALe?** query returns the current scale value for the selected function.

**Return Format**

<scale value><NL>

<scale value> ::= integer in NR1 format

**See Also**

- "Introduction to :**FUNCtion Commands**" on page 213
- "**:FUNCtion:RANGe**" on page 218
:FUNCTION:SOURce

(see page 564)

Command Syntax
:FUNCTION:SOURce <value>

<value> ::= (CHANnel<n> | ADD | SUBTract | MULTiply)

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

The :FUNCTION:SOURce command is only used when an FFT (Fast Fourier Transform), DIFF, or INT operation is selected (see the :FUNCTION:OPERation command for more information about selecting an operation). The :FUNCTION:SOURce command selects the source for function operations. Choose CHANnel<n>, or ADD, SUBT, or MULT to specify the desired source for function DIFFerentiate, INTEGRate, and FFT operations specified by the :FUNCTION:OPERation command.

Query Syntax
:FUNCTION:SOURce?

The :FUNCTION:SOURce? query returns the current source for function operations.

Return Format
<value><NL>

<value> ::= (CHAN<n> | ADD | SUBT | MULT)

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

See Also

- "Introduction to :FUNCTION Commands" on page 213
- ":FUNCTION:OPERation" on page 217
3 Commands by Subsystem

:FUNCtion:SPAN

(see page 564)

Command Syntax

:FUNCtion:SPAN <span>

<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

If you set the frequency span to a value outside of the legal range, the step size is automatically set to the nearest legal value.

The :FUNCtion:SPAN command sets the frequency span of the display (left graticule to right graticule) when FFT (Fast Fourier Transform) is selected.

Query Syntax

:FUNCtion:SPAN?

The :FUNCtion:SPAN? query returns the current frequency span in Hertz.

NOTE

After an *RST (Reset) or :AUToscale command, the values returned by the :FUNCtion:CENTer? and :FUNCtion:SPAN? queries depend on the current :TIMebase:RANGe value. Once you change either the :FUNCtion:CENTer or :FUNCtion:SPAN value, they no longer track the :TIMebase:RANGe value.

Return Format

<span><NL>

<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.

See Also

- "Introduction to :FUNCtion Commands" on page 213
- ":FUNCtion:CENTer" on page 214
- ":TIMebase:RANGe" on page 324
- ":TIMebase:SCALe" on page 327
**:FUNCtion:WINDow**

(see page 564)

**Command Syntax**

:FUNCtion:WINDow <window>

<window> ::= {RECTangular | HANNing | FLATtop}

- The RECTangular window is useful for transient signals, and signals where there are an integral number of cycles in the time record.
- The HANNing window is 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.
- The FLATtop window is best for making accurate amplitude measurements of frequency peaks.

The :FUNCtion:WINDow command allows the selection of three 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.

**Query Syntax**

:FUNCtion:WINDow?

The :FUNCtion:WINDow? query returns the value of the window selected for the FFT function.

**Return Format**

<window><NL>

<window> ::= (RECT | HANN | FLAT)

**See Also**

- "Introduction to :FUNCtion Commands" on page 213
### :HARDcopy Commands

Set and query the selection of hardcopy device and formatting options. See "Introduction to :HARDcopy Commands" on page 224.

#### Table 55 :HARDcopy Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:HARDcopy:FACTors {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:HARDcopy:FFEed {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:HARDcopy:FILename &lt;string&gt; (see page 227)</td>
<td>:HARDcopy:FILename? (see page 227)</td>
<td>&lt;string&gt; ::= quoted ASCII string</td>
</tr>
<tr>
<td>:HARDcopy:FORMat &lt;format&gt; (see page 228)</td>
<td>:HARDcopy:FORMat? (see page 228)</td>
<td>&lt;format&gt; ::= {BMP[24bit]</td>
</tr>
<tr>
<td>:HARDcopy:IGColors {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:HARDcopy:PALETTE &lt;palette&gt; (see page 230)</td>
<td>:HARDcopy:PALETTE? (see page 230)</td>
<td>&lt;palette&gt; ::= {COLOR</td>
</tr>
<tr>
<td>:HARDcopy:PDRiver &lt;driver&gt; (see page 231)</td>
<td>:HARDcopy:PDRiver? (see page 231)</td>
<td>&lt;driver&gt; ::= {AP2XX</td>
</tr>
</tbody>
</table>

#### Introduction to :HARDcopy Commands

The HARDcopy subsystem provides commands to set and query the selection of hardcopy device and formatting options such as inclusion of instrument settings (FACTors) and generation of formfeed (FFEed).

#### Reporting the Setup

Use :HARDcopy? to query setup information for the HARDcopy subsystem.

#### Return Format

The following is a sample response from the :HARDcopy? query. In this case, the query was issued following the *RST command.

```
:HARD:FORM BMP;FIL 'print_00';PDR POST;FACT 0;FFE 0;IGC 0;PAL COL
```
**:HARDcopy:FACTors**

(see page 564)

**Command Syntax**

:HARDcopy:FACTors <factors>

<factors> ::= {{OFF | 0} | {ON | 1}}

The HARDcopy:FACTors command controls whether the scale factors are output on the hardcopy dump.

**Query Syntax**

:HARDcopy:FACTors?

The :HARDcopy:FACTors? query returns a flag indicating whether scale factors are output on the hardcopy.

**Return Format**

<factors><NL>

<factors> ::= {0 | 1}

**See Also**

- "Introduction to :HARDcopy Commands" on page 224
:HAR Dcopy:FFEed

Command Syntax

: H ARDcopy:FFEed <ffeed>

<ffeed> ::= {{OFF | 0} | {ON | 1}}

The HARDcopy:FFEed command controls whether a formfeed is output at the end of a hardcopy dump.

ON (or 1) is only valid when PRINTER0 or PRINTER1 is set as the :HAR Dcopy:FORMAT type.

Query Syntax

: H ARDcopy:FFEed?

The :HARDcopy:FFEed? query returns a flag indicating whether a formfeed is output at the end of the hardcopy dump.

Return Format

<ffeed><NL>

<ffeed> ::= {0 | 1}

See Also

- "Introduction to :HARDcopy Commands" on page 224
- ":HARDcopy:FORMAT" on page 228
:HARDcopy:FILename

(see page 564)

Command Syntax

:HARDcopy:FILename <string>

<string> ::= quoted ASCII string

The HARDcopy:FILename command sets the output filename for those print formats whose output is a file.

Query Syntax

:HARDcopy:FILename?

The :HARDcopy:FILename? query returns the current hardcopy output filename.

Return Format

<string><NL>

<string> ::= quoted ASCII string

See Also

• "Introduction to :HARDcopy Commands" on page 224
• ":HARDcopy:FORMat" on page 228
:*HARDcopy:FORMat*

(see page 564)

**Command Syntax**

```
:HARDcopy:FORMat <format>
```

**Query Syntax**

```
:HARDcopy:FORMat?
```

**Return Format**

```
<format><NL>
```

**See Also**

- "Introduction to :HARDcopy Commands" on page 224
Commands by Subsystem

:HARDcopy:IGColors

(see page 564)

**Command Syntax**

:HARDcopy:IGColors <value>

<value> ::= {OFF | 0} | {ON | 1})

The HARDcopy:IGColors command controls whether the graticule colors are inverted or not.

**Query Syntax**

:HARDcopy:IGColors?

The :HARDcopy:IGColors? query returns a flag indicating whether graticule colors are inverted or not.

**Return Format**

<value><NL>

<value> ::= {0 | 1}

**See Also**

- "Introduction to :HARDcopy Commands" on page 224
3 Commands by Subsystem

:HARDcopy:PALette

Command Syntax

:HARDcopy:PALette <palette>

<palette> ::= (COLOR | GRAYscale)

The HARDcopy:PALette command sets the hardcopy palette color.

Query Syntax

:HARDcopy:PALette?

The :HARDcopy:PALette? query returns the selected hardcopy palette color.

Return Format

<palette><NL>

<palette> ::= (COL | GRAY)

See Also

- "Introduction to :HARDcopy Commands" on page 224
**Commands by Subsystem**

### :HARDcopy:PDRiver

(see page 564)

#### Command Syntax

```
:HARDcopy:PDRiver <driver>
```

<driver> ::= {AP2Xxx | AP21xx | {AP2560 | AP25} | {DJ350 | DJ35} |
            DJ6xx | {DJ630 | DJ63} | DJ6Special | DJ6Photo |
            DJ8Special | DJ8xx | DJ9Vip | OJPRokx50 | DJ9xx | GVIP |
            DJ55xx | {PS470 | PS47} | {PS100 | PS10} | CLASer |
            MLASer | LJFastraster | POSTscript}

The HARDcopy:PDRiver command sets the hardcopy printer driver used for the selected printer.

If the correct driver for the selected printer can be identified, it will be selected and cannot be changed.

#### Query Syntax

```
:HARDcopy:PDRiver?
```

The :HARDcopy:PDRiver? query returns the selected hardcopy printer driver.

#### Return Format

```
<driver><NL>
```

<driver> ::= {AP2X | AP21 | AP25 | DJ35 | DJ6 | DJ63 | DJ6S | DJ6P |
            DJ8S | DJ8 | DJ9V | OJPR | DJ9 | GVIP | DJ55 | PS10 |
            PS47 | CLAS | MLAS | LJF | POST}

#### See Also

- "Introduction to :HARDcopy Commands" on page 224
- ":HARDcopy:FORMat" on page 228
### :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 233.

**Table 56: :MARKer Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MARKer:MODE &lt;mode&gt; (see page 234)</td>
<td>:MARKer:MODE? (see page 234)</td>
<td>&lt;mode&gt; ::= (OFF</td>
</tr>
<tr>
<td>:MARKer:X1Position &lt;position&gt;[suffix] (see page 235)</td>
<td>:MARKer:X1Position? (see page 235)</td>
<td>&lt;position&gt; ::= X1 cursor position value in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>:MARKer:X1Y1source &lt;source&gt; (see page 236)</td>
<td>:MARKer:X1Y1source? (see page 236)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MARKer:X2Position &lt;position&gt;[suffix] (see page 237)</td>
<td>:MARKer:X2Position? (see page 237)</td>
<td>&lt;position&gt; ::= X2 cursor position value in NR3 format [suffix] ::= (s</td>
</tr>
<tr>
<td>:MARKer:X2Y2source &lt;source&gt; (see page 238)</td>
<td>:MARKer:X2Y2source? (see page 238)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>n/a</td>
<td>:MARKer:XDELta? (see page 239)</td>
<td>&lt;return_value&gt; ::= X cursors delta value in NR3 format</td>
</tr>
<tr>
<td>:MARKer:Y1Position &lt;position&gt;[suffix] (see page 240)</td>
<td>:MARKer:Y1Position? (see page 240)</td>
<td>&lt;position&gt; ::= Y1 cursor position value in NR3 format [suffix] ::= (V</td>
</tr>
</tbody>
</table>
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 NONE;X2Y2 NONE;MODE OFF

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MARKer:Y2Position</td>
<td>:MARKer:Y2Position? (see page 241)</td>
<td>&lt;position&gt; ::= Y2 cursor position value in NR3 format</td>
</tr>
<tr>
<td>&lt;position&gt;[suffix]</td>
<td></td>
<td>[suffix] ::= (V</td>
</tr>
<tr>
<td>(see page 241)</td>
<td>:MARKer:YDELta? (see page 242)</td>
<td>&lt;return_value&gt; ::= Y cursors delta value in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 56 :MARKer Commands Summary (continued)
3 Commands by Subsystem

**:MARKer:MODE**

(see page 564)

**Command Syntax**

`:MARKer:MODE <mode>`

<mode> ::= {OFF | MEASurement | MANual}

The :MARKer:MODE command sets the cursors mode. OFF removes the cursor information from the display. MANual mode enables manual placement of the X and Y cursors. In MEASurement mode the cursors track the most recent measurement.

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.

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.

**Query Syntax**

`:MARKer:MODE?`

The :MARKer:MODE? query returns the current cursors mode.

**Return Format**

<mode><NL>

<mode> ::= {OFF | MEAS | MAN}

See Also

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:X1Y1source" on page 236
- ":MARKer:X2Y2source" on page 238
- ":MEASure:SOURce" on page 270
### :MARKer:X1Position

(see page 564)

**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, sets the X1 cursor position and moves the X1 cursor to the specified value.

**Query Syntax**

```
:MARKer:X1Position?
```

The :MARKer:X1Position? query returns the current X1 cursor position. If the front-panel cursors are off an error is returned. This is functionally equivalent to the obsolete :MEASure:TSTArt command/query.

**Return Format**

```
<position><NL>
```

<position> ::= X1 cursor position in NR3 format

**See Also**

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X2Position" on page 237
- ":MARKer:X1Y1source" on page 236
- ":MARKer:X2Y2source" on page 238
- ":MEASure:TSTArt" on page 517
### :MARKer:X1Y1source

(see page 564)

**Command Syntax**

```
:MARKer:X1Y1source <source>
```

<source> ::= (CHANnel<n> | FUNCtion | MATH)

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

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. Sending a :MARKer:X1Y1source command will put the cursors in the MANual mode (see ":MARKer:MODE" on page 234).

This product does not allow independent settings of the X1Y1 and X2Y2 marker sources. Setting the source for one pair of markers sets the source for the other. If :MARKer:MODE is set to OFF or MANual, setting :MEASure:SOURce to CHANnel<n>, FUNCtion, or MATH 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 | NONE)

**See Also**

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X2Y2source" on page 238
- ":MEASure:SOURce" on page 270
:MARKer:X2Position

(see page 564)

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, sets the X2 cursor position and moves the X2 cursor to the specified value.

Query Syntax

:MARKer:X2Position?

The :MARKer:X2Position? query returns current X2 cursor position. If the front-panel cursors are off an error is returned. This is functionally equivalent to the obsolete :MEASURE:TSTOp command/query.

Return Format

<position><NL>

<position> ::= X2 cursor position in NR3 format

See Also

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X1Position" on page 235
- ":MARKer:X2Y2source" on page 238
- ":MEASURE:TSTOp" on page 518
### :MARKer:X2Y2source

**Command Syntax**

`:MARKer:X2Y2source <source>`

- `<source> ::= (CHANnel<n> | FUNCtion | MATH)`
- `<n> ::= (1 | 2 | 3 | 4)` for the four channel oscilloscope models
- `<n> ::= (1 | 2)` for the two channel oscilloscope models

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. Sending a :MARKer:X2Y2source command puts the cursors in the MANual mode (see ":MARKer:MODE" on page 234).

This product does not allow independent settings of the X1Y1 and X2Y2 marker sources. Setting the source for one pair of markers sets the source for the other. If :MARKer:MODE is set to OFF or MANual, setting :MEASure:SOURce to CHANnel<n>, FUNCtion, or MATH 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**

`:MARKer:X2Y2source? <source><NL>`

- `<source> ::= (CHAN<n> | FUNC | NONE)`

**See Also**

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X1Y1source" on page 236
- ":MEASure:SOURce" on page 270
\textbf{:MARKer:XDELta}  

\begin{itemize}
  \item \texttt{N} (see page 564)
\end{itemize}

\textbf{Query Syntax}  

\texttt{:MARKer:XDELta?}

The \texttt{:MARKer:XDELta?} query returns the value difference between the current X1 and X2 cursor positions.

\[ X_{\text{delta}} = (\text{Value at X2 cursor}) - (\text{Value at X1 cursor}) \]

\textbf{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 \texttt{:MARKer:MODE} to \texttt{MANual} to put the cursors in the front-panel Normal mode.

\textbf{Return Format}  

\texttt{<value><NL>}

\texttt{<value>} ::= difference value in NR3 format.

\textbf{See Also}  

- "\texttt{Introduction to :MARKer Commands}" on page 233
- "\texttt{:MARKer:MODE}" on page 234
- "\texttt{:MARKer:X1Position}" on page 235
- "\texttt{:MARKer:X2Position}" on page 237
- "\texttt{:MARKer:X1Y1source}" on page 236
- "\texttt{:MARKer:X2Y2source}" on page 238
\textbf{:MARKer:Y1Position}

(see page 564)

\textbf{Command Syntax}

:MARKer:Y1Position <position> [suffix]

<position> ::= Y1 cursor position in NR3 format

<suffix> ::= \{mV | V | dB\}

The :MARKer:Y1Position command sets :MARKer:MODE to MANual, sets the Y1 cursor position and moves the Y1 cursor to the specified value.

\textbf{Query Syntax}

:MARKer:Y1Position?

The :MARKer:Y1Position? query returns current Y1 cursor position. If the front-panel cursors are off an error is returned. This is functionally equivalent to the obsolete :MEASure:VSTArt command/query

\textbf{Return Format}

<position><NL>

<position> ::= Y1 cursor position in NR3 format

\textbf{See Also}

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X1Y1source" on page 236
- ":MARKer:X2Y2source" on page 238
- ":MARKer:Y2Position" on page 241
- ":MEASure:VSTArt" on page 523
**:MARKer:**Y2Position

(see page 564)

**Command Syntax**

`:MARKer:Y2Position <position> [suffix]`

<position> ::= Y2 cursor position in NR3 format

<suffix> ::= {mV | V | dB}

The :MARKer:Y2Position command sets :MARKer:MODE to MANual, sets the Y2 cursor position and moves the Y2 cursor to the specified value.

**Query Syntax**

`:MARKer:Y2Position?`

The :MARKer:Y2Position? query returns current Y2 cursor position. If the front-panel cursors are off an error is returned. This is functionally equivalent to the obsolete :MEASure:VSTOp command/query.

**Return Format**

<position><NL>

<position> ::= Y2 cursor position in NR3 format

**See Also**

- "Introduction to :MARKer Commands" on page 233
- ":MARKer:MODE" on page 234
- ":MARKer:X1Y1source" on page 236
- ":MARKer:X2Y2source" on page 238
- ":MARKer:Y1Position" on page 240
- ":MEASure:VSTOp" on page 524
**:MARKer:YDELta**

(see page 564)

**Query Syntax**

`:MARKer:YDELta?`

The :MARKer:YDELta? query returns the value difference between the current Y1 and Y2 cursor positions.

\[ Y_{\text{delta}} = (\text{Value at Y2 cursor}) - (\text{Value at Y1 cursor}) \]

**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 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 233
- ":MARKer:MODE" on page 234
- ":MARKer:X1Y1source" on page 236
- ":MARKer:X2Y2source" on page 238
- ":MARKer:Y1Position" on page 240
- ":MARKer:Y2Position" on page 241
### :MEASure Commands

Select automatic measurements to be made and control time markers. See "Introduction to :MEASure Commands" on page 248.

#### Table 57  :MEASure Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASure:CLEar (see page 250)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>:MEASure:COUNter [&lt;source&gt;] (see page 251)</td>
<td>:MEASure:COUNter? [&lt;source&gt;] (see page 251)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;) for DSO models &lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:DEFine DELay, &lt;delay spec&gt; (see page 252)</td>
<td>:MEASure:DEFine? DELay (see page 253)</td>
<td>&lt;delay spec&gt; ::= &lt;edge_spec1&gt;,&lt;edge_spec2&gt; edge_spec1 ::= [&lt;slope&gt;]&lt;occurrence&gt; edge_spec2 ::= [&lt;slope&gt;]&lt;occurrence&gt; &lt;slope&gt; ::= (+</td>
</tr>
<tr>
<td>:MEASure:DEFine THResholds, &lt;threshold spec&gt; (see page 252)</td>
<td>:MEASure:DEFine? THResholds (see page 253)</td>
<td>&lt;threshold spec&gt; ::= (STANdard)</td>
</tr>
<tr>
<td>:MEASure:DELay [&lt;source1&gt;] [,&lt;source2&gt;] (see page 255)</td>
<td>:MEASure:DELay? [&lt;source1&gt;] [,&lt;source2&gt;] (see page 255)</td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>:MEASure:DUTYcycle [&lt;source&gt;] (see page 257)</td>
<td>:MEASure:DUTYcycle? [&lt;source&gt;] (see page 257)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:FALLtime [&lt;source&gt;] (see page 258)</td>
<td>:MEASure:FALLtime? [&lt;source&gt;] (see page 258)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:FREQuency [&lt;source&gt;] (see page 259)</td>
<td>:MEASure:FREQuency? [&lt;source&gt;] (see page 259)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:NWIDth [&lt;source&gt;] (see page 260)</td>
<td>:MEASure:NWIDth? [&lt;source&gt;] (see page 260)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:OVERshoot [&lt;source&gt;] (see page 261)</td>
<td>:MEASure:OVERshoot? [&lt;source&gt;] (see page 261)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>:MEASure:PERiod [&lt;source&gt;] (see page 263)</td>
<td>:MEASure:PERiod? [&lt;source&gt;] (see page 263)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= waveform period in seconds in NR3 format</td>
</tr>
<tr>
<td>:MEASure:PHASe [&lt;source1&gt;] [,&lt;source2&gt;] (see page 264)</td>
<td>:MEASure:PHASe? [&lt;source1&gt;] [,&lt;source2&gt;] (see page 264)</td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= the phase angle value in degrees in NR3 format</td>
</tr>
<tr>
<td>:MEASure:PREShoot [&lt;source&gt;] (see page 265)</td>
<td>:MEASure:PREShoot? [&lt;source&gt;] (see page 265)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= the percent of preshoot of the selected waveform in NR3 format</td>
</tr>
<tr>
<td>:MEASure:PWIDth [&lt;source&gt;] (see page 266)</td>
<td>:MEASure:PWIDth? [&lt;source&gt;] (see page 266)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= width of positive pulse in seconds in NR3 format</td>
</tr>
<tr>
<td>:MEASure:RISEtime [&lt;source&gt;] (see page 267)</td>
<td>:MEASure:RISEtime? [&lt;source&gt;] (see page 267)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= rise time in seconds in NR3 format</td>
</tr>
<tr>
<td>:MEASure:SDEViation [&lt;source&gt;] (see page 268)</td>
<td>:MEASure:SDEViation? [&lt;source&gt;] (see page 268)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= calculated std deviation in NR3 format</td>
</tr>
</tbody>
</table>
### Table 57: :MEASure Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASure:SHOW (1</td>
<td>ON) (see page 269)</td>
<td>:MEASure:SHOW? (see page 269)</td>
</tr>
<tr>
<td>:MEASure:SOURce [&lt;source1&gt;] [,&lt;source2&gt;] (see page 270)</td>
<td>:MEASure:SOURce? (see page 270)</td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source1,2&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:MEASure:TEDGe? &lt;slope&gt;&lt;occurrence&gt;,[,&lt;source&gt;] (see page 272)</td>
<td>&lt;slope&gt; ::= direction of the waveform</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;occurrence&gt; ::= the transition to be reported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:MEASure:TVALue? &lt;value&gt;, [&lt;slope&gt;]&lt;occurrence&gt;[,&lt;source&gt;] (see page 274)</td>
<td>&lt;value&gt; ::= voltage level that the waveform must cross.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;slope&gt; ::= direction of the waveform when &lt;value&gt; is crossed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;occurrence&gt; ::= transitions reported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= time in seconds of specified voltage crossing in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>Command</td>
<td>Query</td>
<td>Options and Query Returns</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>---------------------------</td>
</tr>
</tbody>
</table>
| :MEASure:VAMPlitude ![source](see page 276) | :MEASure:VAMPlitude? ![source](see page 276) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= the amplitude of the selected waveform in volts in NR3 format |
| :MEASure:VAVerage ![source](see page 277) | :MEASure:VAVerage? ![source](see page 277) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= calculated average voltage in NR3 format |
| :MEASure:VBASe ![source](see page 278) | :MEASure:VBASe? ![source](see page 278) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<base_voltage> ::= voltage at the base of the selected waveform in NR3 format |
| :MEASure:VMAX ![source](see page 279) | :MEASure:VMAX? ![source](see page 279) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= maximum voltage of the selected waveform in NR3 format |
| :MEASure:VMIN ![source](see page 280) | :MEASure:VMIN? ![source](see page 280) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= minimum voltage of the selected waveform in NR3 format |
| :MEASure:VPP ![source](see page 281) | :MEASure:VPP? ![source](see page 281) | <source> ::= (CHANnel<n> | FUNCTION | MATH)  
<n> ::= 1-2 or 1-4 in NR1 format  
<return_value> ::= voltage peak-to-peak of the selected waveform 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.

---

#### Table 57: :MEASure Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>[&lt;source&gt;]</td>
<td>(see page 282)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= calculated dc RMS voltage in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:MEASure:VTIMe? &lt;vtime&gt;,&lt;source&gt;</td>
<td>&lt;vtime&gt; ::= displayed time from trigger in seconds in NR3 format</td>
</tr>
<tr>
<td></td>
<td>(see page 283)</td>
<td>&lt;return_value&gt; ::= voltage at the specified time in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:MEASure:VTOP</td>
<td>:MEASure:VTOP? [&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>(see page 284)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= voltage at the top of the waveform in NR3 format</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>(see page 285)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</td>
</tr>
<tr>
<td>:MEASure:XMIN</td>
<td>:MEASure:XMIN? [&lt;source&gt;]</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>[&lt;source&gt;]</td>
<td>(see page 286)</td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= horizontal value of the maximum in NR3 format</td>
</tr>
</tbody>
</table>
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 delayed time base mode (:TIMebase:MODE WINDow), the oscilloscope will attempt to make the measurement inside the delayed 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, NONE
Commands by Subsystem

:MEASure:CLEar

(see page 564)

Command Syntax

:MEASure:CLEar

This command clears all selected measurements and markers from the screen.

See Also

• "Introduction to :MEASure Commands" on page 248
:MEASure:COUNter

(see page 564)

Command Syntax

:MEASure:COUNter [<source>]

<source> ::= (digital channels) | CHANnel<n>
<digital channels> ::= DIGital0,...,DIGital15 for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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 may be selected for the source.

The counter measurement counts trigger level crossings at the selected trigger slope and displays the results in Hz. The gate time for the measurement is automatically adjusted to be 100 ms or twice the current time window, whichever is longer, up to 1 second. The counter measurement can measure frequencies up to 125 MHz. The minimum frequency supported is 1/(2 X gate time).

The Y cursor shows the the edge threshold level used in the measurement.

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:FREQuency" on page 259
- ":MEASure:CLEar" on page 250
:MEASure:DEFine

(see page 564)

Command Syntax

:MEASure:DEFine <meas_spec>

<meas_spec> ::= (DELay | THResholds)

The :MEASure:DEFine command sets up the definition for measurements by specifying the delta time or threshold values. Changing these values may affect the results of other measure commands. The table below identifies which measurement results that can be affected by redefining the DELay specification or the THResholds values. For example, changing the THResholds definition from the default 10%, 50%, and 90% values may change the returned measurement result.

<table>
<thead>
<tr>
<th>MEASure Command</th>
<th>DELay</th>
<th>THResholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUTYcycle</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>DELay</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FALLtime</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>FREQuency</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>NWIDth</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>OVERshoot</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>PERiod</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>PHASE</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>PREShoot</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>PWIDth</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>RISetime</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>VAVerage</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>VRMS</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

:MEASure:DEFine DELay Command Syntax

:MEASure:DEFine DELay,<delay_spec>

<delay_spec> ::= <edge_spec1>,<edge_spec2>

<edge_spec1> ::= [<slope>]<occurrence>

<edge_spec2> ::= [<slope>]<occurrence>

<slope> ::= (+ | -)

<occurrence> ::= integer
This command defines the behavior of the :MEASure:DELay? query by specifying the start and stop edge to be used. <edge_spec1> specifies the slope and edge number on source1. <edge_spec2> specifies the slope and edge number on source2. The measurement is taken as:

\[
\text{delay} = t(<\text{edge}\_\text{spec2}>) - t(<\text{edge}\_\text{spec1}>)
\]

NOTE

The :MEASure:DELay command and the front-panel delay measurement use an auto-edge selection method to determine the actual edge used for the measurement. The :MEASure:DEFine command has no effect on these delay measurements. The edges specified by the :MEASure:DEFine command only define the edges used by the :MEASure:DELay? query.

:MEASure:DEFine

**THResholds**

**Command Syntax**

:MEASure:DEFine THResholds,<threshold spec>

<threshold spec> ::= {STANdard} \\
| {<threshold mode>,<upper>,<middle>,<lower>}

<threshold mode> ::= {PERCent | ABSolute}

for <threshold mode> = PERCent:

<upper>, <middle>, <lower> ::= A number specifying the upper, middle, and lower threshold percentage values between Vbase and Vtop in NR3 format.

for <threshold mode> = ABSolute:

<upper>, <middle>, <lower> ::= A number specifying the upper, middle, and lower threshold absolute values in NR3 format.

- STANdard threshold specification sets the lower, middle, and upper measurement thresholds to 10%, 50%, and 90% values between Vbase and Vtop.
- Threshold mode PERCent sets the measurement thresholds to any user-defined percentages between 5% and 95% of values between Vbase and Vtop.
- Threshold mode ABSolute sets the measurement thresholds to absolute values. ABSolute thresholds are dependent on channel scaling (:CHANnel<n>:RANGE or ":CHANnel<n>:SCALe" on page 183:CHANnel<n>:SCALe), probe attenuation (:CHANnel<n>:PROBe), and probe units (:CHANnel<n>:UNITs). Always set these values first before setting ABSolute thresholds.

**Query Syntax**

:MEASure:DEFine? <meas_spec>

<meas_spec> ::= (DELay | THResholds)

The :MEASure:DEFine? query returns the current edge specification for the delay measurements setup or the current specification for the thresholds setup.
3 Commands by Subsystem

Return Format

for <meas_spec> = DELay:

{ <edge_spec1> | <edge_spec2> | <edge_spec1>,<edge_spec2> } <NL>

for <meas_spec> = THResholds and <threshold mode> = PERCent:

THR,PERC,<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.

for <meas_spec> = THResholds and <threshold mode> = ABSolute:

THR,ABS,<upper>,<middle>,<lower><NL>

<upper>,<middle>,<lower> ::= A number specifying the upper, middle,
and lower threshold voltages in NR3 format.

for <threshold spec> = STANdard:

THR,PERC,+90.0,+50.0,+10.0

See Also

- "Introduction to :MEASure Commands" on page 248
- ":MEAS:DELa" on page 255
- ":MEAS:SOURce" on page 270
- ":CHAN<n>:RANGe" on page 182
- ":CHAN<n>:SCALe" on page 183
- ":CHAN<n>:PROBe" on page 177
- ":CHAN<n>:UNITs" on page 184
Commands by Subsystem

:MEASure:DELay

(see page 564)

Command Syntax

:MEASure:DELay [<source1>][,<source2>]

<source1>, <source2> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :MEASure:DELay command places the instrument in the continuous measurement mode and starts a delay measurement.

The measurement is taken as:

\[
\text{delay} = t(\text{edge spec 2}) - t(\text{edge spec 1})
\]

where the <edge spec> definitions are set by the :MEASure:DEFine command

NOTE

The :MEASure:DELay command and the front-panel delay measurement differ from the :MEASure:DELay? query. The delay command or front-panel measurement run the delay measurement in auto-edge select mode. In this mode, the user may select the edge polarity, but the instrument will select the edges to use to make the best possible delay measurement. The source1 edge chosen will be the edge that meets the polarity specified and is closest to the trigger reference point. The source2 edge selected will be that edge of the specified polarity that gives the first of the following criteria:

- The smallest positive delay value that is less than source1 period.
- The smallest negative delay that is less than source1 period.
- The smallest absolute value of delay.

The :MEASure:DELay? query will make the measurement using the edges specified by the :MEASure:DEFine command.

Query Syntax

:MEASure:DELay? [<source1>][,<source2>]

The :MEASure:DELay? query measures and returns the delay between source1 and source2. The delay measurement is made from the user-defined slope and edge count of the signal connected to source1, to the defined slope and edge count of the signal connected to source2. Delay measurement slope and edge parameters are selected using the :MEASure:DEFine command.

Return Format

<value><NL>

<value> ::= floating-point number delay time in seconds in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:DEFine" on page 252
• ":MEASURE:PHASE" on page 264
Commands by Subsystem

:MEASure:DUTYcycle

Command Syntax

:MEASure:DUTYcycle [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNCtion | MATH}
<digital channels> ::= DIGital0,..,DIGital15 for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :MEASure:DUTYcycle command installs a screen measurement and starts a 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 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:

\[
duty \text{ cycle} = \left(\frac{+\text{pulse width}}{\text{period}}\right) \times 100
\]

Return Format

<value><NL>

<value> ::= ratio of positive pulse width to period in NR3 format

See Also
- "Introduction to :MEASure Commands" on page 248
- ":MEASure:PERiod" on page 263
- ":MEASure:PWIDth" on page 266
- ":MEASure:SOURce" on page 270

Example Code
- "Example Code" on page 270
**:MEASure:FALLtime**

(see page 564)

**Command Syntax**

**:MEASure:FALLtime [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

The :MEASure:FALLtime command installs a screen measurement and starts a fall-time measurement. For highest measurement accuracy, set the sweep speed as fast as possible, while leaving the falling edge of the waveform on the display. If the optional source parameter is specified, the current source is modified.

**NOTE**

This command is not available if the source is FFT (Fast Fourier Transform).

**Query Syntax**

**:MEASure:FALLtime? [<source>]

The :MEASure:FALLtime? query measures and outputs the fall time of the displayed falling (negative-going) edge closest to the trigger reference. The fall time is determined by measuring the time at the upper threshold of the falling edge, then measuring the time at the lower threshold of the falling edge, and calculating the fall time with the following formula:

\[
\text{fall time} = \text{time at lower threshold} - \text{time at upper threshold}
\]

**Return Format**

<value><NL>

<value> ::= time in seconds between the lower threshold and upper threshold in NR3 format

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:RISetime" on page 267
- ":MEASure:SOURce" on page 270
**:MEASure:FREQuency**

(see page 564)

**Command Syntax**

`:MEASure:FREQuency [<source>]`

- `<source>` ::= (`<digital channels>` | CHANnel<n> | FUNCtion | MATH)
- `<digital channels>` ::= DIGital0,..,DIGital15 for the MSO models
- `<n> ::= {1 | 2 | 3 | 4}` for the four channel oscilloscope models
- `<n> ::= {1 | 2}` for the two channel oscilloscope models

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 = \( \frac{1}{(\text{time at trailing rising edge} - \text{time at leading rising edge})} \)

ELSE frequency = \( \frac{1}{(\text{time at trailing falling edge} - \text{time at leading falling edge})} \)

**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 248
- "MEASure:SOURce" on page 270
- "MEASure:PERiod" on page 263

**Example Code**

- "Example Code" on page 270
:MEASure:NWIDth

(see page 564)

**Command Syntax**

:MEASure:NWIDth [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNCTION | MATH}

digital channels ::= DIGital0,..,DIGital15 for the MSO models

<n> := (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> := (1 | 2) for the two channel oscilloscope models

The :MEASure:NWIDth command installs a screen measurement and starts a negative 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: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:

\[
\text{width} = (\text{time at trailing rising edge} - \text{time at leading falling edge})
\]

**Return Format**

/value/<NL>

/value> ::= negative pulse width in seconds in NR3 format

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:SOURce" on page 270
- ":MEASure:PWIDth" on page 266
- ":MEASure:PERiod" on page 263
:MEASure:OVERshoot

(see page 564)

Command Syntax
:MEASure:OVERshoot [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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:
overshoot = ((Vmax-Vtop) / (Vtop-Vbase)) x 100

For a falling edge:
overshoot = ((Vbase-Vmin) / (Vtop-Vbase)) x 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 248
- ":MEASure:PREShoot" on page 265
- ":MEASure:SOURce" on page 270
- ":MEASure:VMAX" on page 279
3  Commands by Subsystem

- ":MEASure:VTOP" on page 284
- ":MEASure:VBASE" on page 278
- ":MEASure:VMIN" on page 280
:MEASure:PERiod

(see page 564)

Command Syntax

:MEASure:PERiod [<source>]

<source> ::= {<digital channels> | CHANnel<n> | FUNCtion | MATH}
<digital channels> ::= DIGital0,..,DIGital15 for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:NWIDth" on page 260
- ":MEASure:PWIDth" on page 266
- ":MEASure:FREQuency" on page 259

Example Code

- "Example Code" on page 270
3 Commands by Subsystem

:MEASure:PHASe

(see page 564)

Command Syntax

:MEASure:PHASe [<source1>][,<source2>]

<source1>, <source2> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

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:

\[
\text{phase} = \frac{\text{delay}}{\text{period of input 1}} \times 360
\]

Return Format

<value><NL>

<value> ::= the phase angle value in degrees in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- ".:MEASure:DELay" on page 255
- ".:MEASure:PERiod" on page 263
- ".:MEASure:SOURce" on page 270
**:MEASure:PREShoot**

(see page 564)

**Command Syntax**

```plaintext
:MEASure:PREShoot [<source>]
```

<source> ::= {CHANnel<n> |FUNCtion| MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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**

```plaintext
: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:

```
preshoot = ((Vmin-Vbase) / (Vtop-Vbase)) x 100
```

For a falling edge:

```
preshoot = ((Vmax-Vtop) / (Vtop-Vbase)) x 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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VMIN" on page 280
- ":MEASure:VMAX" on page 279
- ":MEASure:VTOP" on page 284
- ":MEASure:VBASE" on page 278
Commands by Subsystem

:MEASURE:PWIDth

(see page 564)

Command Syntax

:MEASURE:PWIDth [<source>]

<source> ::= {<digital channels> | CHANNEL<n> | FUNCTION | MATH}
<digital channels> ::= DIGital0,...,DIGital15 for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASURE:SOURce" on page 270
- ":MEASURE:NWIDth" on page 260
- ":MEASURE:PERiod" on page 263
**:MEASure:RISetime**

(see page 564)

**Command Syntax**

```plaintext
:MEASure: RISetime [<source>]
```

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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**

```plaintext
:MEASure: RISetime? [<source>]
```

The :MEASure:RISetime? query measures and outputs the rise time of the displayed rising (positive-going) edge closest to the trigger reference. For maximum measurement accuracy, set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The rise time is determined by measuring the time at the lower threshold of the rising edge and the time at the upper threshold of the rising edge, then calculating the rise time with the following formula:

\[
\text{rise time} = \text{time at upper threshold} - \text{time at lower threshold}
\]

**Return Format**

```plaintext
<value><NL>
```

<value> ::= rise time in seconds in NR3 format

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:SOURce" on page 270
- ":MEASure:FALLtime" on page 258
# :MEASure:SDEViation

(see page 564)

**Command Syntax**

```
:MEASure:SDEViation [<source>]
```

- `<source>` ::= {CHANnel<n> | FUNCtion | MATH}
- `<n>` ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
- `<n>` ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASure:SOURce" on page 270
:MEASure:SHOW

N  (see page 564)

Command Syntax
:MEASure:SHOW <show>

<show> ::= {1 | ON}

The :MEASure:SHOW command enables markers for tracking measurements on the display. This feature is always on.

Query Syntax
:MEASure:SHOW?

The :MEASure:SHOW? query returns the current state of the markers.

Return Format
<show><NL>

<show> ::= 1

See Also
• "Introduction to :MEASure Commands" on page 248
**MEASURE:SOURce**

(see page 564)

**Command Syntax**

:MEASURE:SOURce <source1>[,<source2>]

<source1>,<source2> ::= {<digital channels> | CHANnel<n> | FUNCtion | MATH}

<digital channels> ::= DIGital0,...,DIGital15 for the MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :MEASURE:SOURce command sets the default sources for measurements. The specified sources are used as the sources for the MEASURE subsystem commands if the sources are not explicitly set with the command. If a source is specified for any measurement, the current source is changed to this new value. If :MARKer:MODE is set to OFF or MANual, setting :MEASURE:SOURce to CHANnel<n>, FUNCtion, or MATH will also set :MARKer:X1Y1source to source1 and :MARKer:X2Y2source to source2.

**Query Syntax**

:MEASURE:SOURce?

The :MEASURE:SOURce? query returns the current source selections. If source2 is not specified, the query returns "NONE" for source2. If all channels are off, the query returns "NONE,NONE". Source2 only applies to :MEASURE:DELay and :MEASURE:PHAse measurements.

**NOTE**

MATH is an alias for FUNCTION. The query will return FUNC if the source is FUNCTION or MATH.

**Return Format**

<source1>,<source2><NL>

<source1>,<source2> ::= {<digital channels> | CHAN<n> | FUNC | NONE}

**See Also:**

- "Introduction to :MEASURE Commands" on page 248
- "MARKer:MODE" on page 234
- "MARKer:X1Y1source" on page 236
- "MARKer:X2Y2source" on page 238
- "MEASURE:DELay" on page 255
- "MEASURE:PHASe" on page 264

**Example Code**

' MEASURE - The commands in the MEASURE subsystem are used to make ' measurements on displayed waveforms.
myScope.WriteString "MEASURE:SOURCE CHANNEL1" ' Source to measure.
myScope.WriteString "MEASURE:FREQUENCY?" ' Query for frequency.
varQueryResult = myScope.ReadNumber ' Read frequency.
MsgBox "Frequency:" + vbCrLf _
Agilent 6000 Series Oscilloscopes Programmer's Reference

Commands by Subsystem

Example program from the start: "VISA COM Example in Visual Basic" on page 614
Commands by Subsystem

:MEASure:TEDGe

(see page 564)

Query Syntax

:MEASure:TEDGe? <slope><occurrence>[,<source>]

<slope> ::= direction of the waveform. A rising slope is indicated by a space or plus sign (+). A falling edge is indicated by a minus sign (-).

<occurrence> ::= the transition to be reported. If the occurrence number is one, the first crossing from the left screen edge is reported. If the number is two, the second crossing is reported, etc.

<source> ::= {<digital channels> | CHANnel<n> | FUNCTION | MATH}

<digital channels> ::= DIGital0,...,DIGital15 for the MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

When the :MEASure:TEDGe query is sent, the displayed signal is searched for the specified transition. The time interval between the trigger event and this occurrence is returned as the response to the query. The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the midpoint threshold in the positive direction. Once this crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified vertical value, or if the waveform does not cross the specified vertical value for the specific number of times in the direction specified.

You can make delay and phase measurements using the MEASure:TEDGe command:

\[
\text{Delay} = \text{time at the nth rising or falling edge of the channel} - \text{time at the same edge of another channel}
\]

\[
\text{Phase} = (\text{delay between channels / period of channel}) \times 360
\]

For an example of making a delay and phase measurement, see ":MEASure:TEDGe Code" on page 273.

If the optional source parameter is specified, the current source is modified.
This query is not available if the source is FFT (Fast Fourier Transform).

Return Format

<value><NL>

<value> ::= time in seconds of the specified transition in NR3 format

:MEASure:TEDGe

' Make a delay measurement between channel 1 and 2.
Dim dblChan1Edge1 As Double
Dim dblChan2Edge1 As Double
Dim dblChan1Edge2 As Double
Dim dblDelay As Double
Dim dblPeriod As Double
Dim dblPhase As Double

' Query time at 1st rising edge on ch1.
myScope.WriteString ":MEASURE:TEDGE? +1, CHAN1"

' Read time at edge 1 on ch 1.
dblChan1Edge1 = myScope.ReadNumber

' Query time at 1st rising edge on ch2.
myScope.WriteString ":MEASURE:TEDGE? +1, CHAN2"

' Read time at edge 1 on ch 2.
dblChan2Edge1 = myScope.ReadNumber

' Calculate delay time between ch1 and ch2.
dblDelay = dblChan2Edge1 - dblChan1Edge1

' Write calculated delay time to screen.
MsgBox "Delay = " + vbCrLf + CStr(dblDelay)

' Make a phase difference measurement between channel 1 and 2.
' Query time at 1st rising edge on ch1.
myScope.WriteString ":MEASURE:TEDGE? +2, CHAN1"

' Read time at edge 2 on ch 1.
dblChan1Edge2 = myScope.ReadNumber

' Calculate period of ch 1.
dblPeriod = dblChan1Edge2 - dblChan1Edge1

' Calculate phase difference between ch1 and ch2.
dblPhase = (dblDelay / dblPeriod) * 360
MsgBox "Phase = " + vbCrLf + CStr(dblPhase)

Example program from the start: "VISA COM Example in Visual Basic" on page 614

See Also

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:TVALEue" on page 274
- ":MEASure:VTIMe" on page 283
3 Commands by Subsystem

:MEASure:TVALue

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 | MATH}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

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 248
- ":MEASure:TEDGe" on page 272
- ":MEASure:VTIMe" on page 283
:MEASure:VAMPLitude

(see page 564)

Command Syntax

:MEASure:VAMPLitude [<source>]

<source> ::= {CHANnel<n> | FUNCTION | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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:

vertical amplitude = Vtop - Vbase

Return Format

<value><NL>

<value> ::= the amplitude of the selected waveform in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VBASE" on page 278
- ":MEASure:VTOP" on page 284
- ":MEASure:VPP" on page 281
:MEASure:VAverage

(see page 564)

Command Syntax

:MEASure:VAverage [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

The :MEASure:VAverage command installs a screen measurement and starts an average value measurement. If the optional source parameter is specified, the current source is modified.

Query Syntax

:MEASure:VAverage? [<source>]

The :MEASure:VAverage? query returns the average value of an integral number of periods of the signal. If at least three edges are not present, the oscilloscope averages all data points.

Return Format

<value><NL>

<value> ::= calculated average value in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:SOURce" on page 270
:MEASURE:VBASE

(see page 564)

Command Syntax

:MEASURE:VBASE [<source>]

<source> ::= {CHANNEL<n> | FUNCTION | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASURE:SOURce" on page 270
- ":MEASURE:VTOP" on page 284
- ":MEASURE:VAMPlitude" on page 276
- ":MEASURE:VMIN" on page 280
:MEASure:VMAX

(see page 564)

Command Syntax

:MEASure:VMAX [<source>]

<source> ::= {CHANnel<n> | FUNCTion | MATH}
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VMIN" on page 280
- ":MEASure:VPP" on page 281
- ":MEASure:VTOP" on page 284
**:MEASure:VMIN**

(see page 564)

**Command Syntax**

**:MEASure:VMIN [<source>]**

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VBASE" on page 278
- ":MEASure:VMAX" on page 279
- ":MEASure:VPP" on page 281
**:MEASure:VPP**

(see page 564)

**Command Syntax**

```
:MEASure:VPP [<source>]
```

```
<source> ::= {CHANnel<n> | FUNCtion | MATH}
```

```
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
```

```
<n> ::= {1 | 2} for the two channel oscilloscope models
```

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:

```
Vpp = Vmax - Vmin
```

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VMAX" on page 279
- ":MEASure:VMIN" on page 280
- ":MEASure:VAMPlitude" on page 276
3 Commands by Subsystem

:MEASure:VRMS

(see page 564)

Command Syntax
:MEASure:VRMS [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :MEASure:VRMS command installs a screen measurement and starts a dc RMS 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:VRMS? [<source>]

The :MEASure:VRMS? query measures and outputs the dc RMS value of the selected waveform. The dc RMS value is measured on an integral number of periods of the displayed signal. If at least three edges are not present, the oscilloscope computes the RMS value on all displayed data points.

Return Format

<value><NL>

<value> ::= calculated dc RMS value in NR3 format

See Also
- "Introduction to :MEASure Commands" on page 248
- ":MEASure:SOURce" on page 270
:MEASURE:VTIME

(see page 283)

**Query Syntax**

:MEASURE:VTIME? <vtime_argument>[,<source>]

<vtime_argument> ::= time from trigger in seconds

<source> ::= {<digital channels> | CHANnel<n> | FUNCTION | MATH}

<digital channels> ::= DIGital0,...,DIGital15 for the MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :MEASURE:VTIME? query returns the value at a specified time on the source specified with :MEASURE:SOURce. The specified time must be on the screen and is referenced to the trigger event. 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> ::= value at the specified time in NR3 format

**See Also**

- "Introduction to :MEASURE Commands" on page 248
- ":MEASURE:SOURce" on page 270
- ":MEASURE:TEDGE" on page 272
- ":MEASURE:TVLUE" on page 274
**:MEASure:VTOP**

(see page 564)

**Command Syntax**

`:MEASure:VTOP [<source>]`

- `<source>` ::= {CHANnel<n> | FUNCTION | MATH}
- `<n>` ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
- `<n>` ::= {1 | 2} for the two channel oscilloscope models

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 248
- ":MEASure:SOURce" on page 270
- ":MEASure:VMAX" on page 279
- ":MEASure:VAMPLitude" on page 276
- ":MEASure:VBASE" on page 278
**:MEASure:XMAX**

(see page 564)

**Command Syntax**

```
:MEASure:XMAX [ <source> ]
```

```
<source> ::= {CHANnel<n> | FUNCtion | MATH}
<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models
```

The :MEASure:XMAX command installs a screen measurement and starts an X-at-Max-Y measurement on the selected window. If the optional source parameter is specified, the current source is modified.

**NOTE**

:MEASure:XMAX is an alias for :MEASure:TMAX.

**Query Syntax**

```
:MEASure:XMAX? [ <source> ]
```

The :MEASure:XMAX? query measures and returns the horizontal axis value at which the maximum vertical value occurs. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

**Return Format**

```
<value><NL>
```

```
<value> ::= horizontal value of the maximum in NR3 format
```

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:XMIN" on page 286
- ":MEASure:TMAX" on page 515
3 Commands by Subsystem

:MEASure:XMIN

(see page 564)

Command Syntax

:MEASure:XMIN [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

The :MEASure:XMIN command installs a screen measurement and starts an X-at-Min-Y measurement on the selected window. If the optional source parameter is specified, the current source is modified.

NOTE

:MEASure:XMIN is an alias for :MEASure:TMIN.

Query Syntax

:MEASure:XMIN? [<source>]

The :MEASure:XMIN? query measures and returns the horizontal axis value at which the minimum vertical value occurs. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

Return Format

<value><NL>

<value> ::= horizontal value of the minimum in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- "MEASure:XMAX" on page 285
- "MEASure:TMIN" on page 516
:POD Commands

Control all oscilloscope functions associated with groups of digital channels. See "Introduction to :POD<n> Commands" on page 287.

Table 58 :POD<n> Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:POD&lt;n&gt;:DISPlay {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 in NR1 format</td>
</tr>
<tr>
<td>:POD&lt;n&gt;:SIZE &lt;value&gt; (see page 289)</td>
<td>:POD&lt;n&gt;:SIZE? (see page 289)</td>
<td>&lt;value&gt; ::= (SMALl</td>
</tr>
<tr>
<td>:POD&lt;n&gt;:THReshold &lt;type&gt;[suffix] (see page 290)</td>
<td>:POD&lt;n&gt;:THReshold? (see page 290)</td>
<td>&lt;n&gt; ::= 1-2 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;type&gt; ::= (CMOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;user defined value&gt; ::= value in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= (V</td>
</tr>
</tbody>
</table>

Introduction to :POD<n> Commands

<n> ::= (1 | 2)

The POD subsystem commands control the viewing and threshold of groups of digital channels.

POD1 ::= D0-D7

POD2 ::= D8-D15

NOTE

These commands are only valid for the MSO models.

Reporting the Setup

Use :POD1? or :POD2? to query setup information for the POD subsystem.

Return Format

The following is a sample response from the :POD1? query. In this case, the query was issued following a *RST command.

:POD1:DISP 0;THR +1.40E+00
 Commands by Subsystem

:POD<n>:DISPlay

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.

NOTE
This command is only valid for the MSO models.

Query Syntax
:POD<n>:DISPlay?

The :POD<n>:DISPlay? query returns the current display setting of the specified group of channels.

Return Format
<display><NL>

<display> ::= {0 | 1}

See Also
- "Introduction to :POD<n> Commands" on page 287
- ":DIGital<n>:DISPlay" on page 188
- ":CHANnel<n>:DISPlay" on page 172
- ":VIEW" on page 137
- ":BLANk" on page 109
- ":STATus" on page 134
**:POD<n>:SIZE**

(see page 564)

**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.

**NOTE**

This command is only valid for the MSO models.

**Query Syntax**

```
:POD<n>:SIZE?
```

The :POD<n>:SIZE? query returns the size setting for the specified group of channels.

**Return Format**

```
<size_value><NL>
```

<size_value> ::= {SMALL | MED | LARG}

**See Also**

- "Introduction to :POD<n> Commands" on page 287
- ":DIGital<n>:SIZE" on page 191
- ":DIGital<n>:POSItion" on page 190
3 Commands by Subsystem

:POD<n>:THReshold

(see page 564)

Command Syntax

:POD<n>:THReshold <type>[<suffix>]

<n> ::= An integer, 1 or 2, is attached as a suffix to the command and defines the group of channels that are affected by the command.

$type$ ::= {CMOS | ECL | TTL | <user defined value>}

$user defined value$ ::= -8.00 to +8.00 in NR3 format

$suffix$ ::= {V | mV | uV}

POD1 ::= D0-D7
POD2 ::= D8-D15
TTL ::= 1.4V
CMOS ::= 2.5V
ECL ::= -1.3V

The :POD<n>:THReshold command sets the threshold for the specified group of channels. The threshold is used for triggering purposes and for displaying the digital data as high (above the threshold) or low (below the threshold).

NOTE

This command is only valid for the MSO models.

Query Syntax

:POD<n>:THReshold?

The :POD<n>:THReshold? query returns the threshold value for the specified group of channels.

Return Format

<threshold><NL>

<threshold> ::= Floating point number in NR3 format

See Also

"Introduction to :POD<n> Commands" on page 287
".DIGital<n>:THReshold" on page 192
".TRIGger[:EDGE]:LEVel" on page 366

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"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
## :SBUS Commands

Control oscilloscope functions associated with the serial decode bus. See "Introduction to :SBUS Commands" on page 293.

Table 59  :SBUS Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:BUSDoctor:ADDRESS &lt;value&gt;</td>
<td>:SBUS:BUSDoctor:ADDRESS ? (see page 294)</td>
<td>&lt;value&gt; ::= &lt;field value&gt;, &lt;field value&gt;, &lt;field value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;field value&gt; ::= integer from 0-255 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;value&gt; ::= (see page 294)</td>
</tr>
<tr>
<td></td>
<td>:SBUS:BUSDoctor:BAUDrate &lt;baudrate&gt; (see</td>
<td>&lt;baudrate&gt; ::= (2500000</td>
</tr>
<tr>
<td></td>
<td>page 295)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>:SBUS:BUSDoctor:BAUDrate? (see page 295)</td>
<td></td>
</tr>
<tr>
<td>:SBUS:BUSDoctor:CHANnel &lt;channel&gt;</td>
<td>:SBUS:BUSDoctor:CHANnel ? (see page 296)</td>
<td>&lt;channel&gt; ::= (A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>:SBUS:BUSDoctor:MODE &lt;mode&gt; (see page 297)</td>
<td>&lt;mode&gt; ::= (ASYNchronous</td>
</tr>
<tr>
<td></td>
<td>:SBUS:BUSDoctor:MODE? (see page 297)</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:ERROR? (see page 298)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:OVERload? (see page 299)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:SBUS:CAN:COUNT:RESET (see</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>page 300)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:TOTAL? (see page 301)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:CAN:COUNT:UTILization? (see page</td>
<td>&lt;percent&gt; ::= floating-point in NR3 format</td>
</tr>
<tr>
<td>page 302)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:SBUS:DISPLAY {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:NUL? (see page 304)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:SBUS:FLEXray:COUNT:RESET (see</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>page 305)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:SYNC? (see page 306)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:SBUS:FLEXray:COUNT:TOTAL? (see page 307)</td>
<td>&lt;frame_count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:SBUS:IIC:ASIZE &lt;size&gt; (see</td>
<td>:SBUS:IIC:ASIZE? (see page 308)</td>
<td>&lt;size&gt; ::= (BIT7</td>
</tr>
<tr>
<td>page 308)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introduction to :SBUS Commands

The :SBUS subsystem commands control the serial decode bus viewing, mode, and other options.

These commands are only valid on 4-channel or 4+16-channel oscilloscope models when a serial decode option has been licensed.

Reporting the Setup

Use :SBUS? to query setup information for the :SBUS subsystem.

Return Format

The following is a sample response from the :SBUS? query. In this case, the query was issued following a *RST command.

:SBUS:DISP 0;MODE IIC

### Table 59 : :SBUS Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SBUS:LIN:PARity {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:SBUS:MODE &lt;mode&gt; (see page 310)</td>
<td>:SBUS:MODE? (see page 310)</td>
<td>&lt;mode&gt; ::= {IIC</td>
</tr>
<tr>
<td>:SBUS:SPI:WIDTH &lt;word_width&gt; (see page 311)</td>
<td>:SBUS:SPI:WIDTH? (see page 311)</td>
<td>&lt;word_width&gt; ::= integer 4-16 in NR1 format</td>
</tr>
</tbody>
</table>
**:SBUS:BUSDoctor:ADDRess**

(see page 564)

**Command Syntax**

:SBUS:BUSDoctor:ADDRess <value>

<value> ::= <field value>, <field value>, <field value>, <field value>

<field value> ::= integer from 0-255 in NR1 format

The :SBUS:BUSDoctor:ADDRess command sets the four byte values that make up the BusDoctor's IP address.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the FlexRay triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

:SBUS:BUSDoctor:ADDRess?

The :SBUS:BUSDoctor:ADDRess? query returns the current BusDoctor IP address byte values.

**Return Format**

<value><NL>

<value> ::= <field value>, <field value>, <field value>, <field value>

<field value> ::= integer from 0-255 in NR1 format

**Errors**

- 

**See Also**

- "Introduction to :SBUS Commands" on page 293
- 

":TRIGger:FLEXray Commands" on page 370
**:SBUS:BUSDoctor:BAUDrate**

(see page 564)

**Command Syntax**

```
:SBUS:BUSDoctor:BAUDrate <baudrate>
```

**<baudrate> ::= {2500000 | 5000000 | 10000000}**

The :SBUS:BUSDoctor:BAUDrate command sets the baud rate for the BusDoctor to 2.5 Mb/s, 5 Mb/s, or 10 Mb/s.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the FlexRay triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

```
:SBUS:BUSDoctor:BAUDrate?
```

The :SBUS:BUSDoctor:BAUDrate? query returns the current BusDoctor baud rate setting.

**Return Format**

```
<baudrate><NL>
```

**<baudrate> ::= {2500000 | 5000000 | 10000000}**

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :SBUS Commands" on page 293
- ":TRIGger:FLEXray Commands" on page 370
**:SBUS:BUSDoctor:CHANnel**

(see page 564)

**Command Syntax**

:SBUS:BUSDoctor:CHANnel <channel>

<channel> ::= (A | B)

The :SBUS:BUSDoctor:BAUDrate command sets the channel that the BusDoctor analyzes/preprocesses.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the FlexRay triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

:SBUS:BUSDoctor:CHANnel?

The :SBUS:BUSDoctor:CHANnel? query returns the current BusDoctor channel setting.

**Return Format**

<channel><NL>

<channel> ::= (A | B)

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :SBUS Commands" on page 293
- ":TRIGger:FLEXray Commands" on page 370
**:SBUS:BUSDoctor:MODE**

(see page 564)

**Command Syntax**

:SBUS:BUSDoctor:MODE <mode>

<mode> ::= {ASYNchronous | SYNChronous | PC}

The :SBUS:BUSDoctor:MODE command sets the operating mode of the BusDoctor:

- ASYNchronous — Oscilloscope controls BusDoctor, asynchronous mode monitoring (LAN connection required).
- SYNChronous — Oscilloscope controls BusDoctor, synchronous mode monitoring (LAN connection required).
- PC — PC running Decosys VISION software controls BusDoctor.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the FlexRay triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

:SBUS:BUSDoctor:MODE?

The :SBUS:BUSDoctor:MODE? query returns the current BusDoctor operating mode setting.

**Return Format**

<mode><NL>

<mode> ::= {ASYN | SYNC | PC}

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :SBUS Commands" on page 293
- ":TRIGger:FLEXray Commands" on page 370
**:SBUS:CAN:COUNt:ERRor**

N  (see page 564)

**Query Syntax**

**:SBUS:CAN:COUNt:ERRor?**

Returns the error frame count.

**Return Format**

<frame_count><NL>

<frame_count> ::= integer in NR1 format

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- ":SBUS:CAN:COUNt:RESet" on page 300
- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:CAN Commands" on page 343
:SBUS:CAN:COUNt:OVERload

Query Syntax  
:SBUS:CAN:COUNt:OVERload?

Returns the overload frame count.

Return Format  
<frame_count><NL>

<frame_count> ::= integer in NR1 format

Errors  
• "-241, Hardware missing" on page 535

See Also  
• ":SBUS:CAN:COUNt:RESet" on page 300
• "Introduction to :SBUS Commands" on page 293
• ":SBUS:MODE" on page 310
• ":TRIGger:CAN Commands" on page 343
:SBUS:CAN:COUN:t:RESet

(see page 564)

Command Syntax

:SBUS:CAN:COUN:t:RESet

Resets the frame counters.

Errors

- "-241, Hardware missing" on page 535

See Also

- ":SBUS:CAN:COUN:t:ERRor" on page 298
- ":SBUS:CAN:COUN:t:OVERload" on page 299
- ":SBUS:CAN:COUN:t:TOTal" on page 301
- ":SBUS:CAN:COUN:t:UTILization" on page 302
- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:CAN Commands" on page 343
:SBUS:CAN:COUNt:TOTal

N  (see page 564)

Query Syntax  :SBUS:CAN:COUNt:TOTal?

Returns the total frame count.

Return Format  <frame_count><NL>

<frame_count> ::= integer in NR1 format

Errors  • "-241, Hardware missing" on page 535

See Also  • ":SBUS:CAN:COUNt:RESet" on page 300

• "Introduction to :SBUS Commands" on page 293

• ":SBUS:MODE" on page 310

• ":TRIGger:CAN Commands" on page 343
3 Commands by Subsystem

:SBUS:CAN:COUNt:UTILization

(see page 564)

Query Syntax :SBUS:CAN:COUNt:UTILization?

Returns the percent utilization.

Return Format <percent><NL>

<percent> ::= floating-point in NR3 format

Errors

- "-241, Hardware missing" on page 535

See Also

- ":SBUS:CAN:COUNt:RESet" on page 300
- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:CAN Commands" on page 343
:SBUS:DISPlay

(see page 564)

Command Syntax
:SBUS:DISPlay <display>

<display> ::= {{1 | ON} | {0 | OFF}}

The :SBUS:DISPlay command turns displaying of the serial decode bus on or off.

NOTE
This command is only valid on 4-channel or 4+16-channel oscilloscope models when a serial decode option has been licensed.

Query Syntax
:SBUS:DISPlay?

The :SBUS: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 535

See Also
- "Introduction to :SBUS Commands" on page 293
- ":CHANnel<n>:DISPlay" on page 172
- ":DIGital<n>:DISPlay" on page 188
- ":POD<n>:DISPlay" on page 288
- ":VIEW" on page 137
- ":BLANk" on page 109
- ":STATus" on page 134
:SBUS:FLEXray:COUNt:NULL

(see page 564)

Query Syntax

:SBUS:FLEXray:COUNt:NULL?

Returns the FlexRay null frame count.

Return Format

<frame_count><NL>

<frame_count> ::= integer in NR1 format

Errors

• "-241, Hardware missing" on page 535

See Also

• ":SBUS:FLEXray:COUNt:RESet" on page 305
• "Introduction to :SBUS Commands" on page 293
• ":SBUS:MODE" on page 310
• ":TRIGger:FLEXray Commands" on page 370
:SBUS:FLEXray:COUNt:RESet

(see page 564)

Command Syntax

:SBUS:FLEXray:COUNt:RESet

Resets the FlexRay frame counters.

Errors

- "-241, Hardware missing" on page 535

See Also

- "SBUS:FLEXray:COUNt:NULL" on page 304
- "SBUS:FLEXray:COUNt:SYNC" on page 306
- "SBUS:FLEXray:COUNt:TOTal" on page 307
- "Introduction to :SBUS Commands" on page 293
- "SBUS:MODE" on page 310
- "TRIGger:FLEXray Commands" on page 370
:SBUS:FLEXray:COUNt:SYNC

(see page 564)

Query Syntax
:SBUS:FLEXray:COUNt:SYNC?

Returns the FlexRay sync frame count.

Return Format
<frame_count><NL>
<frame_count> ::= integer in NR1 format

Errors
- "-241, Hardware missing" on page 535

See Also
- ":SBUS:FLEXray:COUNt:RESet" on page 305
- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:FLEXray Commands" on page 370
**:SBUS:FLEXray:COUNt:TOTal**

(see page 564)

**Query Syntax**

:SBUS:FLEXray:COUNt:TOTal?

Returns the FlexRay total frame count.

**Return Format**

<frame_count><NL>

[frame_count] ::= integer in NR1 format

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- ":SBUS:FLEXray:COUNt:RESSet" on page 305
- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:FLEXray Commands" on page 370
### :SBUS:IIC:ASIZe

#### Command Syntax

```
:SBUS:IIC:ASIZe <size>
```

<size> ::= {BIT7 | BIT8}

The :SBUS: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.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the low-speed IIC and SPI serial decode option (Option LSS) has been licensed.

#### Query Syntax

```
:SBUS:IIC:ASIZe?
```

The :SBUS:IIC:ASIZe? query returns the current IIC address width setting.

#### Return Format

```
<mode><NL>
```

<mode> ::= {BIT7 | BIT8}

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :SBUS Commands" on page 293
- ":TRIGger:IIC Commands" on page 391
:SBUS:LIN:PARity

Command Syntax

:SBUS:LIN:PARity <display>

<display> ::= {{1 | ON} | {0 | OFF}}

The :SBUS: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:LIN:PARity?

The :SBUS: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 535

See Also

- "Introduction to :SBUS Commands" on page 293
- ":TRIGger:LIN Commands" on page 400

NOTE

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.
 Commands by Subsystem

:SBUS:MODE

(see page 564)

Command Syntax

:SBUS:MODE <mode>

<mode> ::= {IIC | SPI | CAN | LIN | FLEXray}

The :SBUS:MODE command determines the decode mode for the serial bus.

NOTE

This command is only valid on 4-channel or 4+16-channel oscilloscope models when a serial decode option has been licensed.

Query Syntax

:SBUS:MODE?

The :SBUS:MODE? query returns the current serial bus decode mode setting.

Return Format

<mode><NL>

<mode> ::= {IIC | SPI | CAN | LIN | FLEX | NONE}

Errors

"-241, Hardware missing" on page 535

See Also

"Introduction to :SBUS Commands" on page 293

":TRIGger:MODE" on page 338

":TRIGger:IIC Commands" on page 391

":TRIGger:SPI Commands" on page 416

":TRIGger:CAN Commands" on page 343

":TRIGger:LIN Commands" on page 400
**:SBUS:SPI:WIDTh**

(see page 564)

**Command Syntax**

:SBUS:SPI:WIDTh <word_width>

<word_width> ::= integer 4-16 in NR1 format

The :SBUS:SPI:WIDTh command determines the number of bits in a word of data for SPI.

**NOTE**

This command is only valid on 4-channel or 4+16-channel oscilloscope models when the low-speed IIC and SPI serial decode option (Option LSS) has been licensed.

**Query Syntax**

:SBUS:SPI:WIDTh?

The :SBUS: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 535

**See Also**

- "Introduction to :SBUS Commands" on page 293
- ":SBUS:MODE" on page 310
- ":TRIGger:SPI Commands" on page 416
3  Commands by Subsystem

:SYSTem Commands

Control basic system functions of the oscilloscope. See "Introduction to
:SYSTem Commands" on page 312.

Table 60  :SYSTem Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:SYSTem:DATE &lt;date&gt; (see page 313)</td>
<td>:SYSTem:DATE? (see page 313)</td>
<td>&lt;date&gt; ::= &lt;year&gt;,&lt;month&gt;,&lt;day&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;year&gt; ::= 4-digit year in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;month&gt; ::= (1,...,12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FEbruary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JULy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OCTober</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;day&gt; ::= (1,..31)</td>
</tr>
<tr>
<td>:SYSTem:DSP &lt;string&gt; (see page 314)</td>
<td>n/a</td>
<td>&lt;string&gt; ::= up to 254 characters as a quoted ASCII string</td>
</tr>
<tr>
<td>n/a</td>
<td>:SYSTem:ERRor? (see page 315)</td>
<td>&lt;error&gt; ::= an integer error code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;error string&gt; ::= quoted ASCII string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See Error Messages (see page 533).</td>
</tr>
<tr>
<td>:SYSTem:LOCK (see page 316)</td>
<td>:SYSTem:LOCK? (see page 316)</td>
<td>&lt;value&gt; ::= {ON</td>
</tr>
<tr>
<td>:SYSTem:SETup &lt;setup_data&gt; (see page 317)</td>
<td>:SYSTem:SETup? (see page 317)</td>
<td>&lt;setup_data&gt; ::= data in IEEE 488.2 # format.</td>
</tr>
<tr>
<td>:SYSTem:TIME &lt;time&gt; (see page 319)</td>
<td>:SYSTem:TIME? (see page 319)</td>
<td>&lt;time&gt; ::= hours,minutes,seconds in NR1 format</td>
</tr>
</tbody>
</table>

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

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 312
- ":SYSTem:TIME" on page 319
3 Commands by Subsystem

:SYSTem:DSP

(see page 564)

Command Syntax

:SYSTem:DSP <string>

<string> ::= quoted ASCII string (up to 254 characters)

The :SYSTem:DSP command writes the quoted string (excluding quotation marks) to a text box in the center of the display. Use :SYSTem:DSP "" to remotely remove the message from the display. (Two sets of quote marks without a space between them creates a NULL string.) Press any menu key to manually remove the message from the display.

See Also

• "Introduction to :SYSTem Commands" on page 312
:SYStem:ERRor

(see page 564)

Query Syntax

:SYSTem:ERRor?

The :SYSTem:ERRor? query outputs the next error number and text from
the error queue. The instrument has an error queue that is 30 errors deep
and operates on a first-in, first-out basis. Repeatedly sending the
:SYSTem:ERRor? query returns the errors in the order that they occurred
until the queue is empty. Any further queries then return zero until
another error occurs.

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 "Error Messages" on page 533.

See Also

- "Introduction to :SYSTem Commands" on page 312
- "*ESR (Standard Event Status Register)" on page 82
- "*CLS (Clear Status)" on page 79
Commands by Subsystem

:SYSTem:LOCK

N (see page 564)

Command Syntax

:SYSTem:LOCK <value>

<value> ::= {(1 | ON) | (0 | OFF)}

The :SYSTem:LOCK command disables the front panel. LOCK ON is the equivalent of sending a local lockout message over GPIB.

Query Syntax

:SYSTem:LOCK?

The :SYSTem:LOCK? query returns the lock status of the front panel.

Return Format

<value><NL>

<value> ::= {1 | 0}

See Also

- "Introduction to :SYSTem Commands" on page 312
**:SYSTem:SETup**

(see page 564)

**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 data in IEEE 488.2 # format

**See Also**

- "Introduction to **:SYSTem Commands" on page 312
- "**:LRN (Learn Device Setup)" on page 85

**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,
' #800002204<setup string><NL>
' where the setup string is 2204 bytes in length.
myScope.WriteString "**:SYSTEM:SETUP?"
varQueryResult = myScope.ReadIEEEBlock(BinaryType_U1)
CheckForInstrumentErrors ' After reading query results.

' Output setup string to a file:
Dim strPath As String
strPath = "c:\scope\config\setup.dat"

' Open file for output.
Close #1 ' If #1 is open, close it.
Open strPath For Binary Access Write Lock Write As #1
Put #1, , varQueryResult ' Write data.
Close #1 ' Close file.

' RESTORE_SYSTEM_SETUP - Read the setup string from a file and
' write it back to the oscilloscope.
Dim varSetupString As Variant
strPath = "c:\scope\config\setup.dat"

' Open file for input.
Open strPath For Binary Access Read As #1
Get #1, , varSetupString ' Read data.
Close #1 ' Close file.
Write setup string back to oscilloscope using ":SYSTEM:SETUP"
' command:
myScope.WriteIEEEBlock ":SYSTEM:SETUP ", varSetupString
CheckForInstrumentErrors

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**:SYSTem:TIME**

(see page 564)

**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 312
- ".:SYSTem:DATE" on page 313


3 Commands by Subsystem

:TIMebase Commands

Control all horizontal sweep functions. See "Introduction to :TIMebase Commands" on page 320.

Table 61 :TIMebase Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TIMebase:MODE &lt;value&gt;</td>
<td>:TIMebase:MODE?</td>
<td>&lt;value&gt; ::= (MAIN</td>
</tr>
<tr>
<td>:TIMebase:POSition &lt;pos&gt;</td>
<td>:TIMebase:POSition?</td>
<td>&lt;pos&gt; ::= time from the trigger event to the display reference point in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:RANGe &lt;range_value&gt;</td>
<td>:TIMebase:RANGe?</td>
<td>&lt;range_value&gt; ::= 5 ns through 500 s in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:REFClock {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:TIMebase:REference {LEFT</td>
<td>CENTER</td>
<td>RIGHT}</td>
</tr>
<tr>
<td>:TIMebase:SCALe &lt;scale_value&gt;</td>
<td>:TIMebase:SCALe?</td>
<td>&lt;scale_value&gt; ::= scale value in seconds in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:VERNier {{0</td>
<td>OFF}</td>
<td>{1</td>
</tr>
<tr>
<td>:TIMebase:WINDow:POSition &lt;pos&gt;</td>
<td>:TIMebase:WINDow:POSition?</td>
<td>&lt;pos&gt; ::= time from the trigger event to the delayed view reference point in NR3 format</td>
</tr>
<tr>
<td>:TIMebase:WINDow:RANGe &lt;range_value&gt;</td>
<td>:TIMebase:WINDow:RANGe?</td>
<td>&lt;range_value&gt; ::= range value in seconds in NR3 format for the delayed window</td>
</tr>
<tr>
<td>:TIMebase:WINDow:SCALe &lt;scale_value&gt;</td>
<td>:TIMebase:WINDow:SCALe?</td>
<td>&lt;scale_value&gt; ::= scale value in seconds in NR3 format for the delayed window</td>
</tr>
</tbody>
</table>

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 (delayed) 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 +1.00E-03;POS +0.0E+00


**:TIMebase:MODE**

(see page 564)

**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 (delayed) time base mode, measurements are made in the delayed 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.

**NOTE**

If a :DIGitize command is executed when the :TIMebase:MODE is not MAIN, the :TIMebase:MODE is set to MAIN.

**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 320
- "*RST (Reset)" on page 89
- ":TIMebase:RANGe" on page 324
- ":TIMebase:POSition" on page 323
- ":TIMebase:REFerence" on page 326

**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"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**:TIMebase:POSition**

(see page 564)

**Command Syntax**

```
:TImebase:POSition <pos>
```

<pos> ::= time in seconds from the trigger to the display reference in NR3 format

The :TIMebase:POSition command sets the time interval between the trigger event and the display reference point on the screen. The display reference point is either left, right, or center and is set with the :TIMebase:REFerence command. The maximum position value depends on the time/division settings.

**NOTE**

This command is an alias for the :TIMebase:DElay command.

**Query Syntax**

```
:TImebase:POSition?
```

The :TIMebase:POSition? query returns the current time from the trigger to the display reference in seconds.

**Return Format**

```
<pos><NL>
```

<pos> ::= time in seconds from the trigger to the display reference in NR3 format

**See Also**

- "Introduction to :TIMebase Commands" on page 320
- ":TIMebase:REFerence" on page 326
- ":TIMebase:RANGe" on page 324
- ":TIMebase:SCALe" on page 327
- ":TIMebase:WINDow:POSition" on page 329
- ":TIMebase:DElay" on page 527
**:TIMebase:RANGE**

(see page 564)

Command Syntax

:TIMebase:RANGE <range_value>

<range_value> ::= 5 ns through 500 s in NR3 format

The :TIMebase:RANGE command sets the full-scale horizontal time in seconds for the main window. The range is 10 times the current time-per-division setting.

Query Syntax

:TIMebase:RANGE?

The :TIMebase:RANGE query returns the current full-scale range value for the main window.

Return Format

<range_value><NL>

<range_value> ::= 5 ns through 500 s in NR3 format

See Also

- "Introduction to :TIMebase Commands" on page 320
- ":TIMebase:MODE" on page 322
- ":TIMebase:SCALe" on page 327
- ":TIMebase:WINDow:RANGe" on page 330

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.
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614
:TIMebase:REFClock

(see page 564)

Command Syntax

:TIMebase:REFClock <value>

$value$ ::= {{1 | ON} | {0 | OFF}}

The :TIMebase:REFClock command enables or disables the 10 MHz REF BNC located on the rear panel of the oscilloscope.

The 10 MHz REF BNC can be used as an input for the oscilloscope's reference clock (instead of the internal 10 MHz reference), or it can be used to output the internal 10 MHz reference clock when synchronizing multiple instruments (see ":ACQuire:RSIGnal" on page 146).

The :TIMebase:REFClock ON command enables the 10 MHz REF BNC and sets the reference signal mode to IN. The :TIMebase:REFClock OFF command disables the 10 MHz REF BNC (the same as setting the reference signal mode to OFF).

Query Syntax

:TIMebase:REFClock?

The :TIMebase:REFClock? query returns the current state of the 10 MHz reference signal mode. A "1" indicates that the 10 MHz REF input is enabled (on), and a "0" indicates that either the 10 MHz REF BNC is disabled (off) or that it is set as an output (by the :ACQuire:RSIGnal command).

Return Format

$<value><NL>$

$value$ ::= {0 | 1}

See Also

*:ACQuire:RSIGnal" on page 146
:TIMebase:REFerence

(see page 564)

Command Syntax

:TIMebase:REFerence <reference>

<reference> ::= {LEFT | CENTer | RIGHT}

The :TIMebase:REFerence command sets the time reference to one division from the left side of the screen, to the center of the screen, or to one division from the right side of the screen. Time reference is the point on the display where the trigger point is referenced.

Query Syntax

:TIMebase:REFerence?

The :TIMebase:REFerence? query returns the current display reference for the main window.

Return Format

<reference><NL>

<reference> ::= {LEFT | CENT | RIGH}

See Also

• "Introduction to :TIMebase Commands" on page 320
• "::TIMEbase:MODE" on page 322

Example Code

' TIME_REFERENCE - Possible values are LEFT and CENTER.
' - LEFT sets the display reference on time division from the left.
' - CENTER sets the display reference to the center of the screen.
  myScope.WriteString "::TIMEBASE:REFERENCE CENTER" ' Set reference to center.

Example program from the start: "VISA COM Example in Visual Basic" on page 614
\texttt{:TIMebase:SCALE}

(see page 564)

**Command Syntax**

\texttt{:TIMebase:SCALE <scale_value>}

\(<\text{scale\_value}\rangle := 500\text{ ps through 50 s in NR3 format}\)

The \texttt{:TIMebase:SCALE} command sets the horizontal scale or units per division for the main window.

**Query Syntax**

\texttt{:TIMebase:SCALE?}

The \texttt{:TIMebase:SCALE?} query returns the current horizontal scale setting in seconds per division for the main window.

**Return Format**

\(<\text{scale\_value}\rangle<\text{NL}\>

\(<\text{scale\_value}\rangle := 500\text{ ps through 50 s in NR3 format}\)

**See Also**

- "Introduction to \texttt{:TIMebase Commands}" on page 320
- "\texttt{:TIMebase:RANGe}" on page 324
- "\texttt{:TIMebase:WINDOW:SCALE}" on page 331
- "\texttt{:TIMebase:WINDOW:RANGe}" on page 330
:TIMebase:VERNier

(see page 564)

**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 320
**:TIMebase:WINDow:POSition**

(see page 564)

**Command Syntax**

```
:TIMebase:WINDow:POSition <pos value>
```

<pos value> ::= time from the trigger event to the delayed view reference point in NR3 format

The :TIMebase:WINDow:POSition command sets the horizontal position in the 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 delayed 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 delayed view.

**Return Format**

```
<value><NL>
```

<value> ::= position value in seconds

**See Also**

- "Introduction to :TIMebase Commands" on page 320
- ":TIMebase:MODE" on page 322
- ":TIMebase:POSition" on page 323
- ":TIMebase:RANGe" on page 324
- ":TIMebase:SCALe" on page 327
- ":TIMebase:WINDow:RANGe" on page 330
- ":TIMebase:WINDow:SCALe" on page 331
**:TIMebase:WINDow:RANGe**

(see page 564)

**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 delayed window. The range is 10 times the current delayed 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 320
- ":TIMebase:RANGe" on page 324
- ":TIMebase:POSition" on page 323
- ":TIMebase:SCALe" on page 327
**:TIMebase:WINdow:SCALe**

(see page 564)

**Command Syntax**

```plaintext
:TIMebase:WINdow:SCALe <scale_value>
```

<scale_value> ::= scale value in seconds in NR3 format

The :TIMebase:WINdow:SCALe command sets the 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**

```plaintext
:TIMebase:WINdow:SCALe?
```

The :TIMebase:WINdow:SCALe? query returns the current delayed window scale setting.

**Return Format**

```
<scale_value><NL>
```

<scale_value> ::= current seconds per division for the delayed window

**See Also**

- "Introduction to :TIMebase Commands" on page 320
- ":TIMebase:RANGe" on page 324
- ":TIMebase:POSition" on page 323
- ":TIMebase:SCALe" on page 327
- ":TIMebase:WINdow:RANGe" on page 330
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 342) 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 338).

:TRIGger Commands

Control the trigger modes and parameters for each trigger type. See:

- "Introduction to :TRIGger Commands" on page 332
- "General :TRIGger Commands" on page 335
- ":TRIGger:CAN Commands" on page 343
- ":TRIGger:DURation Commands" on page 354
- ":TRIGger:EBURst Commands" on page 360
- ":TRIGger[:EDGE] Commands" on page 364
- ":TRIGger:FLEXray Commands" on page 370
- ":TRIGger:GLITch Commands" on page 382 (Pulse Width trigger)
- ":TRIGger:IIC Commands" on page 391
- ":TRIGger:LIN Commands" on page 400
- ":TRIGger:SEQUence Commands" on page 408
- ":TRIGger:SPI Commands" on page 416
- ":TRIGger:TV Commands" on page 425
- ":TRIGger:USB Commands" on page 431
• **CAN (Controller Area Network) triggering** will trigger on CAN version 2.0A and 2.0B signals. Setup consists of connecting the oscilloscope to a CAN signal. Baud rate, signal source, and signal polarity, and type of data to trigger on can be specified. With the automotive CAN and LIN serial decode option (Option ASM), you can also trigger on CAN data and identifier patterns, set the bit sample point, and have the module send an acknowledge to the bus when it receives a valid message.

The CAN and LIN serial decode option (Option ASM) replaces the functionality that was available with the N2758A CAN trigger module for the 54620/54640 Series oscilloscopes.

• **Edge triggering** identifies a trigger by looking for a specified slope and voltage level on a waveform.

• **Nth Edge Burst triggering** lets you trigger on the Nth edge of a burst that occurs after an idle time.

• **Pulse width triggering** (:TRIGger:GLITch commands) sets the oscilloscope to trigger on a positive pulse or on a negative pulse of a specified width.

• **Pattern triggering** identifies a trigger condition by looking for a specified pattern. This pattern is a logical AND combination of the channels.

• **Duration triggering** lets you define a pattern, then trigger on a specified time duration.

• **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. With the automotive CAN and LIN serial decode option (Option ASM), you can also trigger on Frame IDs.

• **Sequence triggering** allows you to trigger the oscilloscope after finding a sequence of events. Defining a sequence trigger requires three steps:
  a Define the event to find before you trigger on the next event. This event can be a pattern, and edge from a single channel, or the combination of a pattern and a channel edge.

  b Define the trigger event. This event can be a pattern, and edge from a single channel, the combination of a pattern and a channel edge, or the nth occurrence of an edge from a single channel.

  c Set an optional reset event. This event can be a pattern, an edge from a single channel, the combination of a pattern and a channel edge, or a timeout value.
• **SPI (Serial Peripheral Interface) triggering** consists of connecting the oscilloscope to a clock, data, 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 32 bits long.

• **TV triggering** is used to capture the complicated waveforms of television equipment. The trigger circuitry detects the vertical and horizontal interval of the waveform and produces triggers based on the TV trigger settings you selected. TV triggering requires greater than \(^0\) division of sync amplitude with any analog channel as the trigger source.

• **USB (Universal Serial Bus) triggering** will trigger on a Start of Packet (SOP), End of Packet (EOP), Reset Complete, Enter Suspend, or Exit Suspend signal on the differential USB data lines. USB Low Speed and Full Speed are supported by this trigger.

• **FlexRay triggering** will, when used with a BusDoctor 2 protocol analyzer and a four-channel mixed-signal oscilloscope with Option FRS, trigger on FlexRay bus frames, times, or errors.

**Reporting the Setup**

Use `:TRIGger?` to query setup information for the TRIGger subsystem.

**Return Format**

The return format for the TRIGger? query varies depending on the current mode. The following is a sample response from the :TRIGger? query. In this case, the query was issued following a *RST command.

```
:TRIG:MODE EDGE;SWE AUTO;NREJ 0;HFR 0;HOLD +60.0000000000000E-09;
:TRIG:EDGE:SOUR CHAN1;LEV +0.00000E+00;SLOP POS;REJ OFF;COUP DC
```
# General :TRIGger Commands

## Table 62  General :TRIGger Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:HFReject {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>:TRIGger:HOLDoff &lt;holdoff_time&gt; (see page 337)</td>
<td>:TRIGger:HOLDoff (see page 337)</td>
<td>&lt;holdoff_time&gt; ::= 60 ns to 10 s in NR3 format</td>
</tr>
<tr>
<td>:TRIGger:MODE &lt;mode&gt; (see page 338)</td>
<td>:TRIGger:MODE? (see page 338)</td>
<td>&lt;mode&gt; ::= {EDGE</td>
</tr>
<tr>
<td>:TRIGger:NREJect {{0</td>
<td>OFF}</td>
<td>(1</td>
</tr>
<tr>
<td>:TRIGger:PATTern &lt;value&gt;, &lt;mask&gt; [,&lt;edge source&gt;,&lt;edge&gt;] (see page 340)</td>
<td>:TRIGger:PATTern? (see page 341)</td>
<td>&lt;value&gt; ::= 32-bit integer or &lt;string&gt; &lt;mask&gt; ::= 32-bit integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnnnnnnn&quot;; n ::= {0,...,9</td>
</tr>
<tr>
<td>:TRIGger:SWEep &lt;sweep&gt; (see page 342)</td>
<td>:TRIGger:SWEep? (see page 342)</td>
<td>&lt;sweep&gt; ::= {AUTO</td>
</tr>
</tbody>
</table>
**:TRIGger:**HFReject

(see page 564)

**Command Syntax**

```plaintext
: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**

```plaintext
:TRIGger:HFReject?
```

The :TRIGger:HFReject? query returns the current high frequency reject filter mode.

**Return Format**

```plaintext
<value><NL>
<value> ::= {0 | 1}
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger[:EDGE]:REJect" on page 367
**:TRIGger:HOLDoff**

(see page 564)

**Command Syntax**

```plaintext
:TRIGger:HOLDoff <holdoff_time>
```

<holdoff_time> ::= 60 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**

```plaintext
:TRIGger:HOLDoff?  
```

The :TRIGger:HOLDoff? query returns the holdoff time value for the current trigger mode.

**Return Format**

```plaintext
<holdoff_time><NL>
```

<holdoff_time> ::= the holdoff time value in seconds in NR3 format.

**See Also**

- "Introduction to :TRIGger Commands" on page 332
**:TRIGGER:MODE**

(see page 564)

**Command Syntax**

`:TRIGGER:MODE <mode>`

<mode> ::= {EDGE | GLITch | PATTern | CAN | DURation | IIC | EBURst
                   | LIN | SEQuence | SPI | TV | USB | FLEXray}

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> ::= {NONE | EDGE | GLIT | PATT | CAN | DUR | IIC
                   | EBUR | LIN | SEQ | SPI | TV | USB | FLEX}

**See Also**

- "Introduction to :TRIGGER Commands" on page 332
- ":TRIGGER:SWEep" on page 342
- ":TIMebase:MODE" on page 322

**Example Code**

' TRIGGER_MODE - Set the trigger mode to EDGE, GLITch, PATTern, CAN,
   DURation, IIC, EBURSt, LIN, SEQuence, SPI, TV, or USB.

   ' Set the trigger mode to EDGE.
   myScope.WriteString "::TRIGGER:MODE EDGE"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
**:TRIGger:NREJect**

(see page 564)

Command Syntax

```
:TRIGger:NREJect <value>
```

```
<value> ::= {{0 | OFF} | {1 | ON}}
```

The :TRIGger:NREJect command turns the noise reject filter off and on. When the noise reject filter is on, the trigger circuitry is less sensitive to noise but may require a greater amplitude waveform to trigger the oscilloscope. This command is not valid in TV trigger mode.

Query Syntax

```
:TRIGger:NREJect?
```

The :TRIGger:NREJect? query returns the current noise reject filter mode.

Return Format

```
<value>NL
```

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

See Also

- "Introduction to :TRIGger Commands" on page 332
:TRIGger:PATTern

(see page 564)

Command Syntax

:TRIGger:PATTern <pattern>

<pattern> ::= <value>, <mask> [, <edge source>, <edge>]

<value> ::= 32-bit integer or <string>

<mask> ::= 32-bit integer or <string>

<string> ::= "0xnnnnnn"; n ::= {0,..,9 | A,..,F}

<edge source> ::= {CHANnel<n> | EXTernal | NONE} for DSO models

<edge source> ::= {CHANnel<n> | DIGital0,..,DIGital15 | NONE} for MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

<edge> ::= {POSitive | NEGative}

The :TRIGger:PATTern command defines the specified pattern resource according to the value and the mask. For both <value> and <mask>, each bit corresponds to a possible trigger channel. The bit assignments vary by instrument:

<table>
<thead>
<tr>
<th>Oscilloscope Models</th>
<th>Value and Mask Bit Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 1 through 4.</td>
</tr>
<tr>
<td>2 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - analog channels 1 and 2.</td>
</tr>
<tr>
<td>4 analog channels only</td>
<td>Bits 0 through 3 - analog channels 1 through 4. Bit 4 - external trigger.</td>
</tr>
<tr>
<td>2 analog channels only</td>
<td>Bits 0 and 1 - analog channels 1 and 2. Bit 4 - external trigger.</td>
</tr>
</tbody>
</table>

Set a <value> bit to "0" to set the pattern for the corresponding channel to low. Set a <value> bit to "1" to set the pattern to high.

Set a <mask> bit to "0" to ignore the data for the corresponding channel. Only channels with a "1" set on the appropriate mask bit are used.

**NOTE**

The optional source and the optional edge should be sent together or not at all. The edge will be set in the simple pattern if it is included. If the edge source is also specified in the mask, the edge takes precedence.
Query Syntax  

`:TRIGger:PA TTern?`

The `:TRIGger:PA TTern?` query returns the pattern value, the mask, and the edge of interest in the simple pattern.

Return Format  

`<pattern><NL>`

See Also  

- "Introduction to :TRIGger Commands" on page 332
- " :TRIGger:MODE" on page 338
3 Commands by Subsystem

:TRIGger:SWEep

(see page 564)

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 332
### :TRIGGER:CAN Commands

**Table 63 :TRIGGER:CAN Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :TRIGGER:CAN:_PATTERN:DATA `<value>`, `<mask>` (see page 345) | :TRIGGER:CAN:_PATTERN:DATA? (see page 345) | `<value>` ::= 64-bit integer in decimal, `<nondecimal>`, or `<string>` (with Option AMS)  
<mask> ::= 64-bit integer in decimal, `<nondecimal>`, or `<string>`  
<nondex> ::= #Hnn...n where n ::= {0,..,9 | A,...,F} for hexadecimal  
<nondex> ::= #Bnn...n where n ::= {0 | 1} for binary  
<string> ::= "0xnn...n" where n ::= {0,..,9 | A,...,F} for hexadecimal |
| :TRIGGER:CAN:_PATTERN:DATA_LENGTH `<length>` (see page 346) | :TRIGGER:CAN:_PATTERN:DATA_LENGTH? (see page 346) | `<length>` ::= integer from 1 to 8 in NR1 format (with Option AMS) |
| :TRIGGER:CAN:_PATTERN:ID `<value>`, `<mask>` (see page 347) | :TRIGGER:CAN:_PATTERN:ID? (see page 347) | `<value>` ::= 32-bit integer in decimal, `<nondecimal>`, or `<string>` (with Option AMS)  
<mask> ::= 32-bit integer in decimal, `<nondecimal>`, or `<string>`  
<nondex> ::= #Hnn...n where n ::= {0,..,9 | A,...,F} for hexadecimal  
<nondex> ::= #Bnn...n where n ::= {0 | 1} for binary  
<string> ::= "0xnn...n" where n ::= {0,..,9 | A,...,F} for hexadecimal |
| :TRIGGER:CAN:_PATTERN:ID_MODE `<value>` (see page 348) | :TRIGGER:CAN:_PATTERN:ID_MODE? (see page 348) | `<value>` ::= {STANDARD | EXTENDED} (with Option AMS) |
| :TRIGGER:CAN:_SAMPLEPOINT `<value>` (see page 349) | :TRIGGER:CAN:_SAMPLEPOINT? (see page 349) | `<value>` ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format |
| :TRIGGER:CAN:_SIGNAL:BAUDRATE `<baudrate>` (see page 350) | :TRIGGER:CAN:_SIGNAL:BAUDRATE? (see page 350) | `<baudrate>` ::= {10000 | 20000 | 33300 | 50000 | 62500 | 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000} |
### Commands by Subsystem

**Table 63: :TRIGger:CAN Commands Summary (continued)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:CAN:SOURce</td>
<td>:TRIGger:CAN:SOURce?</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;condition&gt; ::= (SOF</td>
</tr>
</tbody>
</table>
**:TRIGger:CAN:PATTern:DATA**

(see page 564)

**Command Syntax**

`:TRIGger:CAN:PATTern:DATA <value>,<mask>`

- `<value>` ::= 64-bit integer in decimal, <nondecimal>, or <string>
- `<mask>` ::= 64-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

The :TRIGger:CAN:PATTern:DATA command defines the CAN data pattern resource according to the value and the mask. This pattern, along with the data length (set by the :TRIGger:CAN:PATTern:DATA:LENGth command), control the data pattern searched for in each CAN message.

Set a `<value>` bit to "0" to set the corresponding bit in the data pattern to low. Set a `<value>` bit to "1" to set the bit to high.

Set a `<mask>` bit to "0" to ignore that bit in the data stream. Only bits with a "1" set on the mask are used.

**NOTE**

If more bytes are sent for `<value>` or `<mask>` than specified by the :TRIGger:CAN:PATTern:DATA:LENGth command, the most significant bytes will be truncated. If the data length is changed after the `<value>` and `<mask>` are programmed, the added or deleted bytes will be added to or deleted from the least significant bytes.

**NOTE**

This command is only valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

**Query Syntax**

`:TRIGger:CAN:PATTern:DATA?`


**Return Format**

`<value>, <mask><NL>` in nondecimal format

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 346
- ":TRIGger:CAN:PATTern:ID" on page 347
:TRIGger:CAN:PATTern:DATA:LENGth

(see page 564)

Command Syntax

:TRIGger:CAN:PATTern:DATA:LENGth <length>

<length> ::= integer from 1 to 8 in NR1 format

The :TRIGger:CAN: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 0 bytes to 8 bytes (64 bits). The value for these bytes is set by the :TRIGger:CAN:PATTern:DATA command.

NOTE

This command is only valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

Query Syntax

:TRIGger:CAN:PATTern:DATA:LENGth?

The :TRIGger:CAN: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 535

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:CAN:PATTern:DATA" on page 345
- ":TRIGger:CAN:SOURce" on page 351
:TRIGger:CAN:PARTtern:ID

(see page 564)

**Command Syntax**

```
:TRIGger:CAN:PARTtern:ID <value>, <mask>
```

<value> ::= 32-bit integer in decimal, <nondecimal>, or <string>

<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

The :TRIGger:CAN:PARTtern:ID command defines the CAN identifier pattern resource according to the value and the mask. This pattern, along with the identifier mode (set by the :TRIGger:CAN:PARTtern:ID:MODE command), control the identifier pattern searched for in each CAN message.

Set a <value> bit to "0" to set the corresponding bit in the identifier pattern to low. Set a <value> bit to "1" to set the bit to high.

Set a <mask> bit to "0" to ignore that bit in the identifier stream. Only bits with a "1" set on the mask are used.

**NOTE**

If more bits are sent than allowed (11 bits in standard mode, 29 bits in extended mode) by the :TRIGger:CAN:PARTtern:ID:MODE command, the most significant bytes will be truncated. If the ID mode is changed after the <value> and <mask> are programmed, the added or deleted bits will be added to or deleted from the most significant bits.

**NOTE**

This command is only valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

**Query Syntax**

```
:TRIGger:CAN:PARTtern:ID?
```

The :TRIGger:CAN:PARTtern:ID? query returns the current settings of the specified CAN identifier pattern resource.

**Return Format**

<value>, <mask><NL> in nondecimal format

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:CAN:PARTtern:ID:MODE" on page 348
- ":TRIGger:CAN:PARTtern:DATA" on page 345
:TRIGGER:CAN:PATTERN:ID:MODE

(see page 564)

Command Syntax

:TRIGGER:CAN:PATTERN:ID:MODE <value>

,value> ::= {STANDARD | EXTENDED}


NOTE

This command is only valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

Query Syntax

:TRIGGER:CAN:PATTERN:ID:MODE?

The :TRIGGER:CAN: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 535

See Also

- "Introduction to :TRIGGER Commands" on page 332
- ":TRIGGER:MODE" on page 338
- ":TRIGGER:CAN:PATTERN:DATA" on page 345
- ":TRIGGER:CAN:PATTERN:DATA:LENGTH" on page 346
- ":TRIGGER:CAN:PATTERN:ID" on page 347
**:TRIGger:CAN:SAMPlepoint**

(see page 564)

### Command Syntax

```plaintext
:TRIGger:CAN:SAMPlepoint <value>

<value><NL>

<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
```

The :TRIGger: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

```plaintext
:TRIGger:CAN:SAMPlepoint?
```

The :TRIGger:CAN:SAMPlepoint? query returns the current CAN sample point setting.

### Return Format

```plaintext
<value><NL>

<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
```

### See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:TRIGger" on page 352
:TRIGger:CAN:SIGNal:BAUDrate

Command Syntax
:TRIGger:CAN:SIGNal:BAUDrate <baudrate>

<baudrate> ::= integer in NR1 format
<baudrate> ::= {10000 | 20000 | 33300 | 50000 | 62500 | 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000}

The :TRIGger:CAN:SIGNal:BAUDrate command sets the standard baud rate of the CAN signal from 10 kb/s to 1 Mb/s. If a non-standard baud rate is sent, the baud rate will be set to the next highest standard rate.

If the baud rate you select does not match the system baud rate, false triggers may occur.

Query Syntax
:TRIGger:CAN:SIGNal:BAUDrate?

The :TRIGger:CAN:SIGNal:BAUDrate? query returns the current CAN baud rate setting.

Return Format
<baudrate><NL>
<baudrate> ::= integer = {10000 | 20000 | 33300 | 50000 | 62500 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000}

See Also
- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:TRIGger" on page 352
- ":TRIGger:CAN:SIGNal:DEFinition" on page 529
- ":TRIGger:CAN:SOURce" on page 351
**:TRIGger:CAN:SOURce**

N (see page 564)

**Command Syntax**

**:TRIGger:CAN:SOURce <source>**

- `<source>` ::= (CHANnel<n> | EXTernal) for the DSO models
- `<source>` ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models
- `<n>` ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
- `<n>` ::= (1 | 2) for the two channel oscilloscope models

The :TRIGger:CAN:SOURce command sets the source for the CAN signal. The source setting is only valid when :TRIGger:CAN:TRIGger is set to SOF (start of frame).

**Query Syntax**

**:TRIGger:CAN:SOURce?**

The :TRIGger:CAN:SOURce? query returns the current source for the CAN signal.

**Return Format**

```
<source><NL>
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:TRIGger" on page 352
- ":TRIGger:CAN:SIGNal:DEFinition" on page 529
**:TRIGger:CAN:TRIGger**

(see page 564)

**Command Syntax**

`:TRIGger:CAN:TRIGger <condition>`

<condition> ::= {SOF | DATA | ERrOr | IDData | IDEither | IDRemote | ALLerrors | OVERload | ACKerror}

The "**:TRIGger:CAN:TRIGger**" command sets the CAN trigger on condition:

- **SOF** - will trigger on the Start of Frame (SOF) bit of a Data frame, Remote Transfer Request (RTR) frame, or an Overload frame.
- **DATA** - will trigger on CAN Data frames matching the specified Id, Data, and the DLC (Data length code).
- **ERRor** - will trigger on CAN Error frame.
- **IDData** - will trigger on CAN frames matching the specified Id of a Data frame.
- **IDEither** - will trigger on the specified Id, regardless if it is a Remote frame or a Data frame.
- **IDRemote** - will trigger on CAN frames matching the specified Id of a Remote frame.
- **ALLerrors** - will trigger on CAN active error frames and unknown bus conditions.
- **OVERload** - will trigger on CAN overload frames.
- **ACKerror** - will trigger on a data or remote frame acknowledge bit that is recessive.

The table below shows the programming parameter and the corresponding front-panel softkey selection:

<table>
<thead>
<tr>
<th>Remote &lt;condition&gt; parameter</th>
<th>Front-panel Trigger on: softkey selection (softkey text - softkey popup text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF</td>
<td>SOF - Start of Frame</td>
</tr>
<tr>
<td>DATA</td>
<td>Id &amp; Data - Data Frame Id and Data</td>
</tr>
<tr>
<td>ERrOr</td>
<td>Error - Error frame</td>
</tr>
<tr>
<td>IDData</td>
<td>Id &amp; ~RTR - Data Frame Id (~RTR)</td>
</tr>
<tr>
<td>IDEither</td>
<td>Id - Remote or Data Frame Id</td>
</tr>
<tr>
<td>IDRemote</td>
<td>Id &amp; RTR - Remote Frame Id (RTR)</td>
</tr>
<tr>
<td>ALLerrors</td>
<td>All Errors - All Errors</td>
</tr>
<tr>
<td>OVERload</td>
<td>Overload - Overload Frame</td>
</tr>
<tr>
<td>ACKerror</td>
<td>Ack Error - Acknowledge Error</td>
</tr>
</tbody>
</table>

CAN Data specification is set by the :TRIGger:CAN:PATTern:DATA command.

CAN Data Length Code is set by the :TRIGger:CAN:PATTern:DATA:LENGth command.

**NOTE**

SOF is the only valid selection for analog oscilloscopes. If the automotive CAN and LIN serial decode option (Option AMS) has not been licensed, SOF is the only valid selection.

**Query Syntax**

:TRIGger:CAN:TRIGger?

The :TRIGger:CAN:TRIGger? query returns the current CAN trigger on condition.

**Return Format**

<condition><NL>

<condition> ::= {SOF | DATA | ERR | IDD | IDE | IDR | ALL | OVER | ACK}

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:PATTern:DATA" on page 345
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 346
- ":TRIGger:CAN:PATTern:ID" on page 347
- ":TRIGger:CAN:PATTern:ID:MODE" on page 348
- ":TRIGger:CAN:SIGNal:DEFinition" on page 529
- ":TRIGger:CAN:SOURce" on page 351
### :TRIGger:DURation Commands

#### Table 64 :TRIGger:DURation Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:DURation:GREaterthan &lt;greater than</td>
<td>:TRIGger:DURation:GREaterthan? (see page</td>
<td>&lt;greater than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td>time&gt;[suffix] (see page 355)</td>
<td>355)</td>
<td>[suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:DURation:LESSthan &lt;less than time&gt;</td>
<td>:TRIGger:DURation:LESSthan? (see page 356)</td>
<td>&lt;less than time&gt; ::= floating-point number from 5 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td>[suffix] (see page 356)</td>
<td></td>
<td>[suffix] ::= (s</td>
</tr>
<tr>
<td>:TRIGger:DURation:PATTERN &lt;value&gt;, &lt;mask&gt;</td>
<td>:TRIGger:DURation:PATTern? (see page 357)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt;</td>
</tr>
<tr>
<td>(see page 357)</td>
<td></td>
<td>&lt;mask&gt; ::= integer or &lt;string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;string&gt; ::= &quot;0xnnnnnn&quot; n ::= (0,..,9</td>
</tr>
<tr>
<td>:TRIGger:DURation:QUALifier &lt;qualifier&gt;</td>
<td>:TRIGger:DURation:QUALifier? (see page 358)</td>
<td>&lt;qualifier&gt; ::= (GREaterthan</td>
</tr>
<tr>
<td>(see page 358)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:TRIGger:DURation:RANGE &lt;greater than time&gt;</td>
<td>:TRIGger:DURation:RANGE? (see page 359)</td>
<td>&lt;greater than time&gt; ::= min duration from 10 ns to 9.99 seconds in NR3 format</td>
</tr>
<tr>
<td>[suffix], &lt;less than time&gt;[suffix] (see page</td>
<td></td>
<td>&lt;less than time&gt; ::= max duration from 15 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td>359)</td>
<td></td>
<td>[suffix] ::= (s</td>
</tr>
</tbody>
</table>
:TRIGger:DURation:GREaterthan

(see page 564)

Command Syntax

:TRIGger:DURation:GREaterthan <greater than time>[<suffix>]

<greater than time> ::= minimum trigger duration in seconds
(5 ns - 10 seconds) in NR3 format

<suffix> ::= {s | ms | us | ns | ps }

The :TRIGger:DURation:GREaterthan command sets the minimum duration for the defined pattern when :TRIGger:DURation:QUALifier is set to GREaterthan. The command also sets the timeout value when the :TRIGger:DURation:QUALifier is set to TIMeout.

Query Syntax

:TRIGger:DURation:GREaterthan?

The :TRIGger:DURation: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 332
- ":TRIGger:DURation:PATTern" on page 357
- ":TRIGger:DURation:QUALifier" on page 358
- ":TRIGger:MODE" on page 338
**:TRIGger:DURation:LESSthan**

(see page 564)

**Command Syntax**

```
:TRIGger:DURation:LESSthan <less than time>[<suffix>]
```

<less than time> ::= maximum trigger duration in seconds
(5 ns - 10 seconds) in NR3 format

<suffix> ::= {s | ms | us | ns | ps}

The :TRIGger:DURation:LESSthan command sets the maximum duration for
the defined pattern when :TRIGger:DURation:QUALifier is set to LESSthan.

**Query Syntax**

```
:TRIGger:DURation:LESSthan?
```

The :TRIGger:DURation:LESSthan? query returns the duration time for the
defined pattern.

**Return Format**

```
<less than time><NL>
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:DURation:PATTern" on page 357
- ":TRIGger:DURation:QUALifier" on page 358
- ":TRIGger:MODE" on page 338
**:TRIGger:DURation:PATTern**

(see page 564)

**Command Syntax**

`:TRIGger:DURation:PATTern <value>, <mask>`

<value> ::= integer or <string>

<mask> ::= integer or <string>

<string> ::= "0xnnnnnn"; n ::= {0,..,9, A,..,F}

The :TRIGger:DURation:PATTern command defines the specified duration pattern resource according to the value and the mask. For both <value> and <mask>, each bit corresponds to a possible trigger channel. The bit assignments vary by instrument:

<table>
<thead>
<tr>
<th>Oscilloscope Models</th>
<th>Value and Mask Bit Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 1 through 4.</td>
</tr>
<tr>
<td>2 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - analog channels 1 and 2.</td>
</tr>
<tr>
<td>4 analog channels only</td>
<td>Bits 0 through 3 - analog channels 1 through 4. Bit 4 - external trigger.</td>
</tr>
<tr>
<td>2 analog channels only</td>
<td>Bits 0 and 1 - analog channels 1 and 2. Bit 4 - external trigger.</td>
</tr>
</tbody>
</table>

Set a <value> bit to "0" to set the pattern for the corresponding channel to low. Set a <value> bit to "1" to set the pattern to high.

Set a <mask> bit to "0" to ignore the data for the corresponding channel. Only channels with a "1" set on the appropriate mask bit are used.

**Query Syntax**

`:TRIGger:DURation:PATTern?`

The :TRIGger:DURation:PATTern? query returns the pattern value.

**Return Format**

<value>, <mask><NL>

<value> ::= a 32-bit integer in NR1 format.

<mask> ::= a 32-bit integer in NR1 format.

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ".:TRIGger:PATTern" on page 340
**:TRIGger:DURation:QUALifier**

(see page 564)

**Command Syntax**

:TRIGger:DURation:QUALifier <qualifier>

<qualifier> ::= {GREaterthan | LESSthan | INRange | OUTRange | TIMeout}

The :TRIGger:DURation:QUALifier command qualifies the trigger duration.

Set the GREaterthan qualifier value with the :TRIGger:DURation:GREaterthan command.

Set the LESSthan qualifier value with the :TRIGger:DURation:LESSthan command.

Set the INRange and OUTRange qualifier values with the :TRIGger:DURation:RANGe command.

Set the TIMeout qualifier value with the :TRIGger:DURation:GREaterthan command.

**Query Syntax**

:TRIGger:DURation:QUALifier?

The :TRIGger:DURation:QUALifier? query returns the trigger duration qualifier.

**Return Format**

<qualifier><NL>

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- "TRIGger:DURation:GREaterthan" on page 355
- "TRIGger:DURation:LESSthan" on page 356
- "TRIGger:DURation:RANGe" on page 359
:TRIGger:DURation:RANGe

(see page 564)

Command Syntax

:TRIGger:DURation:RANGe <greater than time>[<suffix>],
<less than time>[<suffix>]

<greater than time> ::= minimum duration in seconds
(10 ns - 9.99 seconds) in NR3 format

<less than time> ::= maximum duration in seconds
(15 ns - 10 seconds) in NR3 format

<suffix> ::= {s | ms | us | ns | ps}

The :TRIGger:DURation:RANGe command sets the duration for the defined pattern when the :TRIGger:DURation:QUALifier command is set to INRange or OUTRange.

NOTE

If you set the minimum duration longer than the maximum duration, the order of the parameters is automatically reversed.

Query Syntax

:TRIGger:DURation:RANGe?

The :TRIGger:DURation:RANGe? query returns the duration time for the defined pattern.

Return Format

<greater than time>,<less than time><NL>

See Also

• "Introduction to :TRIGger Commands" on page 332
• ":TRIGger:DURation:PATTern" on page 357
• ":TRIGger:DURation:QUALifier" on page 358
• ":TRIGger:MODE" on page 338
The :TRIGger:EDGE:SOURce command is used to specify the source channel for the Nth Edge Burst trigger. If an analog channel is selected as the source, the :TRIGger:EDGE:LEVel command is used to set the Nth Edge Burst trigger level. If a digital channel is selected as the source, the :DIGital<n>:THReshold or :POD<n>:THReshold command is used to set the Nth Edge Burst trigger level.
**:TRIgger:EBURst:COUNt**

(see page 564)

**Command Syntax**

```
:TRIgger:EBURst:COUNt <count>
```

<count> ::= integer in NR1 format

The :TRIgger:EBURst:COUNt command sets the Nth edge at burst counter resource. The edge counter is used in the trigger stage to determine which edge in a burst will generate a trigger.

**Query Syntax**

```
:TRIgger:EBURst:COUNt?
```

The :TRIgger:EBURst:COUNt? query returns the current Nth edge of burst edge counter setting.

**Return Format**

```
<count><NL>
```

<count> ::= integer in NR1 format

**See Also**

- "Introduction to :TRIgger Commands" on page 332
- ":TRIgger:EBURst:SLOPe" on page 363
- ":TRIgger:EBURst:IDLE" on page 362
**:TRIGger:EBURst:IDLE**

(see page 564)

**Command Syntax**

`:TRIGger:EBURst:IDLE <time_value>`

<time_value> ::= time in seconds in NR3 format

The :TRIGger:EBURst:IDLE command sets the Nth edge in a burst idle resource in seconds from 10 ns to 10 s. The timer is used to set the minimum time before the next burst.

**Query Syntax**

`:TRIGger:EBURst:IDLE?`


**Return Format**

<time_value><NL>

<time_value> ::= time in seconds in NR3 format

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:EBURst:SLOPe" on page 363
- ":TRIGger:EBURst:COUNT" on page 361
**:TRIGger:EBURst:SLOPe**

N \[\text{(see page 564)}\]

**Command Syntax**

**:TRIGger:EBURst:SLOPe <slope>**

<slope> ::= (NEGative | POSitive)

The :TRIGger:EBURst:SLOPe command specifies whether the rising edge (POSitive) or falling edge (NEGative) of the Nth edge in a burst will generate a trigger.

**Query Syntax**

**:TRIGger:EBURst:SLOPe?**

The :TRIGger:EBURst:SLOPe? query returns the current Nth edge in a burst slope.

**Return Format**

<slope><NL>

<slope> ::= (NEG | POS)

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ".:TRIGger:EBURst:IDLE" on page 362
- ".:TRIGger:EBURst:COUNT" on page 361
### :TRIGger[:EDGE] Commands

#### Table 66  :TRIGger[:EDGE] Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger[:EDGE]:COUPlin g {AC</td>
<td>DC</td>
<td>LF} (see page 365)</td>
</tr>
</tbody>
</table>
| :TRIGger[:EDGE]:LEVel <level> [,<source>] (see page 366) | :TRIGger[:EDGE]:LEVel? [<source>] (see page 366) | For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format.  
For external triggers, <level> ::= 2 volts with probe attenuation at 1:1 in NR3 format.  
For digital channels (MSO models), <level> ::= 8 V.  
<source> ::= {CHANnel<n> | EXTernal} for DSO models  
<source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | EXTernal } for MSO models  
<n> ::= 1-2 or 1-4 in NR1 format |
| :TRIGger[:EDGE]:REJect {OFF | LF | HF} (see page 367) | :TRIGger[:EDGE]:REJect? (see page 367) | {OFF | LF | HF} |
| :TRIGger[:EDGE]:SLOPe <polarity> (see page 368) | :TRIGger[:EDGE]:SLOPe? (see page 368) | <polarity> ::= {POSitive | NEGative | ETHer | ALTernate} |
| :TRIGger[:EDGE]:SOURce <source> (see page 369) | :TRIGger[:EDGE]:SOURce? (see page 369) | <source> ::= {CHANnel<n> | EXTernal} for DSO models  
<source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | EXTernal} for MSO models  
<n> ::= 1-2 or 1-4 in NR1 format |
Commands by Subsystem

**:TRIGger[:EDGE]:COUPling**

(see page 564)

**Command Syntax**

```plaintext
:TRIGger[:EDGE]:COUPling <coupling>
```

```plaintext
<coupling> ::= {AC | DC | LFR}
```

The :TRIGger[:EDGE]:COUPling command sets the input coupling for the selected trigger sources. The coupling can be set to AC, DC, or LFR.

- 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**

```plaintext
:TRIGger[:EDGE]:COUPling?
```

The :TRIGger[:EDGE]:COUPling? query returns the current coupling selection.

**Return Format**

```plaintext
<coupling><NL>
```

```plaintext
<coupling> ::= {AC | DC | LFR}
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ".:TRIGger:MODE" on page 338
- ".:TRIGger[:EDGE]:REJect" on page 367
### :TRIgger[:EDGE]:LEVel

(see page 564)

**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>` ::= 2 V with probe attenuation at 1:1 in NR3 format for external triggers
- `<level>` ::= 8 V for digital channels (MSO models)
- `<source>` ::= {CHANnel<n> | EXTernal} for the DSO models
- `<source>` ::= {CHANnel<n> | DIGital0,...,DIGital15 | EXTernal} for the MSO models
- `<n>` ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
- `<n>` ::= (1 | 2) for the two channel oscilloscope models

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 332
- ":TRIgger[:EDGE]:SOURce" on page 369
- ":POD<n>:THReshold" on page 290
- ":DIGital<n>:THReshold" on page 192
:TRIGGER[:EDGE]:REJECT

(see page 564)

Command Syntax

:TRIGGER[:EDGE]:REJECT <reject>

<reject> ::= {OFF | LFR| HFR}

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 332
- ":TRIGGER:HFR|ect" on page 336
- ":TRIGGER[:EDGE]:COUPLing" on page 365
3 Commands by Subsystem

:TRIGger[:EDGE]:SLOPe

(see page 564)

Command Syntax

:TRIGger[:EDGE]:SLOPe <slope>

<slope> ::= {NEGative | POSitive | EITHer | ALTernate}

The :TRIGger[:EDGE]:SLOPe command specifies the slope of the edge for the trigger. The SLOPe command is not valid in TV trigger mode. Instead, use :TRIGger:TV:POLarity to set the polarity in TV trigger mode.

Query Syntax

:TRIGger[:EDGE]:SLOPe?

The :TRIGger[:EDGE]:SLOPe? query returns the current trigger slope.

Return Format

<slope><NL>

<slope> ::= {NEG | POS | EITH | ALT}

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:TV:POLarity" on page 428

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"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
:TRIGger[:EDGE]:SOURce

Command Syntax

:TRIGger[:EDGE]:SOURce <source>

<source> ::= (CHANnel<n> | EXTernal | LINE) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15 | EXTernal | LINE)
for the MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models

The :TRIGger[:EDGE]:SOURce command selects the channel that produces
the trigger.

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> | EXT | LINE | NONE) for the DSO models
<source> ::= (CHAN<n> | DIG0,...,DIG15 | EXTernal | LINE | NONE)
for the MSO models

See Also

* "Introduction to :TRIGger Commands" on page 332
* ":TRIGger:MODE" on page 338

Example Code

' TRIGGER_EDGE_SOURCE - Selects the channel that actually produces th
edge trigger. Any channel can be selected.
myScope.WriteString ':TRIGGER:EDGE:SOURCE CHANNEL1'

Example program from the start: "VISA COM Example in Visual Basic" on
page 614
## :TRIGger:FLEXray Commands

### Table 67 :TRIGger:FLEXray Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:FLEXray:TIME:SEGment &lt;segment_type&gt;</td>
<td>:TRIGger:FLEXray:TIME:SEGment?</td>
<td>&lt;segment_type&gt; ::= {STATic</td>
</tr>
</tbody>
</table>
:TRIGger:FLEXray:ERRor:TYPE

(see page 564)

Command Syntax

:TRIGger:FLEXray:ERRor:TYPE <error_type>

<error_type> ::= {ALL | CODE | TSS | HCRC | FCRC | END | BOUNdary | IDLE | SYMbol | SLOT | NULL | SOS | FID | CCOunt | PLENght}

Selects the FlexRay error type to trigger on. The error type setting is only valid when the FlexRay trigger mode is set to ERRor.

- ALL – triggers on ALL errors.
- CODE – triggers on only CODE errors (NRZ).
- TSS – triggers on only TSS violations.
- HCRC – triggers on only Header CRC errors.
- FCRC – triggers on only Frame CRC errors.
- END – triggers on only frame END sequence errors.

The following error types are NOT valid when the BUSDoctor is in ASYNcronous mode:

- BOUNdary – triggers on only BOUNdary violations.
- IDLE – triggers only on Network IDLE time violations.
- SYMbol – triggers only on SYMbol window violations.
- SLOT – triggers only on SLOT overbooked errors.
- NULL – triggers only on NULL frame errors.
- SOS – triggers only on Sync Or Startup errors.
- FID – triggers only on Frame ID errors.
- CCOunt – triggers only on Cycle Count errors.
- PLENght – triggers only on static Payload LENgth errors.

NOTE

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

Query Syntax

:TRIGger:FLEXray:ERRor:TYPE?

The :TRIGger:FLEXray:ERRor:TYPE? query returns the currently selected FLEXray error type.

Return Format

<error_type><NL>

<error_type> ::= {ALL | CODE | TSS | HCRC | FCRC | END | BOUNdary | IDLE | SYMbol | SLOT | NULL | SOS | FID | CCOunt | PLENght}
Errors

- "-241, Hardware missing" on page 535

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:FLEXray:TRIGger" on page 381
Commands by Subsystem

:TRIgger:FLEXray:FRAMe:CCBase

(see page 564)

**Command Syntax**

```plaintext
:TRIgger:FLEXray:FRAMe:CCBase <cycle_count_base>
```

<cycle_count_base> ::= integer from 0-63

The :TRIgger:FLEXray:FRAMe:CCBase command sets the base of the FlexRay cycle count (in the frame header) to trigger on. The cycle count base setting is only valid when the FlexRay trigger mode is set to FRAME.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

```plaintext
:TRIgger:FLEXray:FRAMe:CCBase? 
```

The :TRIgger:FLEXray:FRAMe:CCBase? query returns the current cycle count base setting for the FlexRay frame trigger setup.

**Return Format**

```plaintext
<cycle_count_base><NL>
```

<cycle_count_base> ::= integer from 0-63

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIgger Commands" on page 332
- ":TRIgger:FLEXray:TRIgger" on page 381
**:TRIGger:FLEXray:FRAME:CCRepetition**

(see page 564)

**Command Syntax**

`:TRIGger:FLEXray:FRAME:CCRepetition <cycle_count_repetition>`

`<cycle_count_repetition> ::= {ALL | <rep #>}`

`<rep #> ::= integer from 2-64`

The :TRIGger:FLEXray:FRAME:CCRepetition command sets the repetition number of the FlexRay cycle count (in the frame header) to trigger on. The cycle count repetition setting is only valid when the FlexRay trigger mode is set to FRAME.

**Note**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

`:TRIGger:FLEXray:FRAME:CCRepetition?`

The :TRIGger:FLEXray:FRAME:CCRepetition? query returns the current cycle count repetition setting for the FlexRay frame trigger setup.

**Return Format**

`<cycle_count_repetition><NL>`

`<cycle_count_repetition> ::= {ALL | <rep #>}`

`<rep #> ::= integer from 2-64`

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:FLEXray:TRIGger" on page 381
Commands by Subsystem

:TRIGger:FLEXray:FRAME:ID

Command Syntax

:TRIGger:FLEXray:FRAME:ID <frame_id>

<frame_id> ::= {ALL | <frame #>}

<frame #> ::= integer from 1-2047

The :TRIGger:FLEXray:FRAME:ID command sets the FlexRay frame ID to trigger on. The frame IF setting is only valid when the FlexRay trigger mode is set to FRAME.

NOTE

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

Query Syntax

:TRIGger:FLEXray:FRAME:ID?

The :TRIGger:FLEXray:FRAME:ID? query returns the current frame ID setting for the FlexRay frame trigger setup.

Return Format

[frame_id]<NL>

[frame_id] ::= {ALL | <frame #>}

[frame #] ::= integer from 1-2047

Errors

- "-241, Hardware missing" on page 535

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":.TRIGger:MODE" on page 338
- ":.TRIGger:FLEXray:TRIGger" on page 381
**:TRIGger:FLEXray:FRAMe:TYPE**

(see page 564)

**Command Syntax**

`:TRIGger:FLEXray:FRAMe:TYPE <frame_type>`

`<frame_type> ::= {NORM | STAR | NULL | SYNC | NSTA | NNUL | NSYN}`

The :TRIGger:FLEXray:FRAMe:TYPE command sets the FlexRay frame type to trigger on. The frame type setting is only valid when the FlexRay trigger mode is set to FRAME.

- **NORMal** — will trigger on only normal (NSTA & NNUL & NSYN) frames.
- **STARtup** — will trigger on only startup frames.
- **NULL** — will trigger on only null frames.
- **SYNC** — will trigger on only sync frames.
- **NSTA** — will trigger on frames other than startup frames.
- **NNUL** — will trigger on frames other than null frames.
- **NSYN** — will trigger on frames other than sync frames.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

`:TRIGger:FLEXray:FRAMe:TYPE?`

The :TRIGger:FLEXray:FRAMe:TYPE? query returns the current frame type setting for the FlexRay frame trigger setup.

**Return Format**

`<frame_type><NL>`

`<frame_type> ::= {NORM | STAR | NULL | SYNC | NSTA | NNUL | NSYN}`

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:FLEXray:TRIGger" on page 381
**:TRIGger:FLEXray:TIME:CBASe**

(see page 564)

**Command Syntax**

`:TRIGger:FLEXray:TIME:CBASe <cycle_base>`

<cycle_base> ::= integer from 0-63

The :TRIGger:FLEXray:TIME:CBASe command sets the base of the FlexRay cycle to trigger on. The cycle base setting is only valid when the FlexRay trigger mode is set to TIME.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

`:TRIGger:FLEXray:TIME:CBASe?`

The :TRIGger:FLEXray:TIME:CBASe? query returns the current cycle base setting for the FlexRay time trigger setup.

**Return Format**

<cycle_base><NL>

<cycle_base> ::= integer from 0-63

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:FLEXray:TRIGger" on page 381
**:TRIGger:FLEXray:TIME:CREPetition**

(see page 564)

**Command Syntax**

:TRIGger:FLEXray:TIME:CREPetition <cycle_repetition>

<cycle_repetition> ::= {ALL | <rep #>}

<rep #> ::= integer from 2-64

The :TRIGger:FLEXray:TIME:CREPetition command sets the repetition number of the FlexRay cycle to trigger on. The cycle repetition setting is only valid when the FlexRay trigger mode is set to TIME.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

:TRIGger:FLEXray:TIME:CREPetition?

The :TRIGger:FLEXray:TIME:CREPetition? query returns the current cycle repetition setting for the FlexRay time trigger setup.

**Return Format**

<cycle_repetition><NL>

<cycle_repetition> ::= {ALL | <rep #>}

<rep #> ::= integer from 2-64

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:FLEXray:TRIGger" on page 381
**Commands by Subsystem 3**

---

**:TRIGger:FLEXray:TIME:SEGMen**

(see page 564)

**Command Syntax**

`:TRIGger:FLEXray:TIME:SEGMen <segment_type>`

<segment_type> ::= {STATic | DYNamic | SYMbol | IDLE}

The :TRIGger:FLEXray:TIME:SEGMen command sets the FlexRay segment type. The segment setting is only valid when the FlexRay trigger mode is set to TIME.

---

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

---

**Query Syntax**

`:TRIGger:FLEXray:TIME:SEGMen?`

The :TRIGger:FLEXray:TIME:SEGMen? query returns the current segment setting for the FlexRay time trigger setup.

**Return Format**

<segment_type><NL>

<segment_type> ::= {STAT | DYN | SYM | IDLE}

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- "::TRIGger:MODE" on page 338
- "::TRIGger:FLEXray:TRIGger" on page 381
**:TRIGger:FLEXray:TIME:SLOT**

(see page 564)

**Command Syntax**

```
:TRIGger:FLEXray:TIME:SLOT <slot_type>, <slot_id>
```

- `<slot_type>` ::= {ALL | EMPTY}
- `<slot_id>` ::= {ALL | <slot #>}
- `<slot #>` ::= integer from 1-2047

The :TRIGger:FLEXray:TIME:SLOT command sets the FlexRay slot type and ID. The slot setting is only valid when the FlexRay trigger mode is set to TIME.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

**Query Syntax**

```
:TRIGger:FLEXray:TIME:SLOT?
```

The :TRIGger:FLEXray:TIME:SLOT? query returns the current source for the FLEXray signal.

**Return Format**

```
<slot_type>, <slot_id><NL>
```

- `<slot_type>` ::= {ALL | EMPTY}
- `<slot_id>` ::= {ALL | <slot #>}
- `<slot #>` ::= integer from 1-2047

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:FLEXray:TRIGger" on page 381
Commands by Subsystem

:TRIGger:FLEXray:TRIGger

(see page 564)

Command Syntax

:TRIGger:FLEXray:TRIGger <condition>

<condition> ::= (FRAME | TIME | ERRor)

The :TRIGger:FLEXray:TRIGger command sets the FLEXray trigger on condition:

- **FRAME** — triggers on specified frames (without errors).
- **TIME** — triggers on specified bus cycles, segments, and slots.
- **ERRor** — triggers on selected active error frames and unknown bus conditions.

**NOTE**

This command is only valid when the FLEXray triggering and serial decode option (Option FRS) has been licensed.

Query Syntax

:TRIGger:FLEXray:TRIGger?

The :TRIGger:FLEXray:TRIGger? query returns the current FLEXray trigger on condition.

Return Format

<condition><NL>

<condition> ::= (FRAM | TIME | ERR)

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:FLEXray:ERRor:TYPE" on page 371
- ":TRIGger:FLEXray:FRAME:CCBase" on page 373
- ":TRIGger:FLEXray:FRAME:CCRepetition" on page 374
- ":TRIGger:FLEXray:FRAME:ID" on page 375
- ":TRIGger:FLEXray:FRAME:TYPE" on page 376
- ":TRIGger:FLEXray:TIME:CBASe" on page 377
- ":TRIGger:FLEXray:TIME:CREPetition" on page 378
- ":TRIGger:FLEXray:TIME:SEGMent" on page 379
- ":TRIGger:FLEXray:TIME:SLOT" on page 380
### :TRIGger:GLITch Commands

**Table 68 :TRIGger:GLITch Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :TRIGger:GLITch:GREATER than <greater than time>[suffix] (see page 384) | :TRIGger:GLITch:GREATER than? (see page 384) | <greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format<br>
[suffix] ::= {s | ms | us | ns | ps} |
| :TRIGger:GLITch:LESS THAN <less than time>[suffix] (see page 385) | :TRIGger:GLITch:LESS THAN? (see page 385) | <less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format<br>
[suffix] ::= {s | ms | us | ns | ps} |
| :TRIGger:GLITch:LEVEL <level> [<source>] (see page 386) | :TRIGger:GLITch:LEVEL? (see page 386) | For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format.<br>
For external triggers, <level> ::= 2 volts with probe attenuation at 1:1 in NR3 format.<br>
For digital channels (MSO models),<br>
<level> ::= 6 V.<br>
<source> ::= (CHANnel<n> | EXTernal) for DSO models<br>
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for MSO models<br>
<n> ::= 1-2 or 1-4 in NR1 format |
| :TRIGger:GLITch:POLARITY <polarity> (see page 387) | :TRIGger:GLITch:POLARITY? (see page 387) | <polarity> ::= (POSitive | NEGative) |
| :TRIGger:GLITch:QUALIFIER <qualifier> (see page 388) | :TRIGger:GLITch:QUALIFIER? (see page 388) | <qualifier> ::= (GREATERthan | LESSthan | RANGE) |
Table 68  :TRIGger:GLITch Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:GLITch:RANGE &lt;greater than time&gt;[suffix], &lt;less than time&gt;[suffix] (see page 389)</td>
<td>:TRIGger:GLITch:RANGE? (see page 389)</td>
<td>&lt;greater than time&gt; ::= start time from 10 ns to 9.99 seconds in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;less than time&gt; ::= stop time from 15 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[suffix] ::= {s</td>
</tr>
<tr>
<td>:TRIGger:GLITch:SOURce &lt;source&gt; (see page 390)</td>
<td>:TRIGger:GLITch:SOURce? (see page 390)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
</tbody>
</table>
:TRIGger:GLITch:GREaterthan
(see page 564)

Command Syntax
:TRIGger:GLITch:GREaterthan <greater_than_time>[<suffix>]

<greater_than_time> ::= 32-bit floating-point number (5 ns - 10 seconds)
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>.

See Also
- "Introduction to :TRIGger Commands" on page 332
- ".:TRIGger:GLITch:SOURce" on page 390
- ".:TRIGger:GLITch:QUALifier" on page 388
- ".:TRIGger:MODE" on page 338
**:TRIGger:GLITch:LESSthan**

(see page 564)

**Command Syntax**

```plaintext
:TRIGger:GLITch:LESSthan <less_than_time>[<suffix>]
```

<less_than_time> ::= floating-point number (5 ns - 10 seconds)

<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**

```plaintext
:TRIGger:GLITch:LESSthan?
```


**Return Format**

```
<less_than_time><NL>
```

<less_than_time> ::= a 32-bit floating-point number in NR3 format.

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:GLITch:SOURce" on page 390
- ":TRIGger:GLITch:QUALifier" on page 388
- ":TRIGger:MODE" on page 338
3 Commands by Subsystem

:TRIGger:GLITch:LEVel

N (see page 564)

Command Syntax

:TRIGger:GLITch:LEVel <level_argument>

[level_argument] ::= [level][, [source]]

[level] ::= .75 x full-scale voltage from center screen in NR3 format for internal triggers

[level] ::= 2 V with probe attenuation at 1:1 in NR3 format for external triggers

[level] ::= 6 V for digital channels (MSO models)

[source] ::= {CHANnel<n> | EXTernal} for DSO models

[source] ::= {CHANnel<n> | DIGital0,...,DIGital15} for MSO models

<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models

<n> ::= (1 | 2) for the two channel oscilloscope models

The :TRIGger:GLITch:LEVel command sets the trigger level voltage for the active pulse width trigger.

Query Syntax

:TRIGger:GLITch:LEVel?

The :TRIGger:GLITch:LEVel? query returns the trigger level of the current pulse width trigger mode. If all channels are off, the query returns "NONE."

Return Format

[level_argument]<NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIG:MODE" on page 338
- ":TRIGger:GLITch:SOURce" on page 390
:TRIGger:GLITch:POLarity

(see page 564)

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 332
• ":TRIGger:MODE" on page 338
• ":TRIGger:GLITch:SOURce" on page 390
**:TRIGger:GLITch:QUALifier**

(see page 564)

**Command Syntax**

```plaintext
: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**

```plaintext
:TRIGger:GLITch:QUALifier?
```

The :TRIGger:GLITch:QUALifier? query returns the glitch pulse width qualifier.

**Return Format**

```plaintext
<operator><NL>
```

<operator> ::= {GRE | LESS | RANG}

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:GLITch:SOURce" on page 390
- ":TRIGger:MODE" on page 338
**:TRIGger:GLITch:RANGe**

(see page 564)

**Command Syntax**

```
:TRIGger:GLITch:RANGe <greater than time>[suffix],
        <less than time>[suffix]
```

<greater than time> ::= start time (10 ns - 9.99 seconds) in NR3 format

<less than time> ::= stop time (15 ns - 10 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. If you set the stop time before the
start time, the order of the parameters is automatically reversed.

**Query Syntax**

```
:TRIGger:GLITch:RANGE?
```

The :TRIGger:GLITch:RANGE? query returns the pulse width duration time
for :TRIGger:GLITch:SOURce.

**Return Format**

```
<start time>,<stop time><NL>
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:GLITch:SOURce" on page 390
- ":TRIGger:GLITch:QUALifier" on page 388
- ":TRIGger:MODE" on page 338
:TRIGger:GLITch:SOURce

Command Syntax
:TRIGger:GLITch:SOURce <source>

<source> ::= (CHANnel<n> | EXternal) for the DSO models
<source> ::= (DIGital0,..,DIGital15 | CHANnel<n>) for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

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 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:GLITch:LEVel" on page 386
- ":TRIGger:GLITch:POLarity" on page 387
- ":TRIGger:GLITch:QUALifier" on page 388
- ":TRIGger:GLITch:RANGe" on page 389

Example Code
- "Example Code" on page 369
### :TRIGger:IIC Commands

#### Table 69 :TRIGger:IIC Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
</table>
| :TRIGger:IIC:PATTern:ADRess <value> (see page 392) | :TRIGger:IIC:PATTern:ADRess? (see page 392) | <value> ::= integer or <string>  
<string> ::= "0xnn" n ::= (0,..,9 | A,..,F) |
| :TRIGger:IIC:PATTern:DATa <value> (see page 393) | :TRIGger:IIC:PATTern:DATa? (see page 393) | <value> ::= integer or <string>  
<string> ::= "0xnn" n ::= (0,..,9 | A,..,F) |
| :TRIGger:IIC:PATTern:DATa2 <value> (see page 394) | :TRIGger:IIC:PATTern:DATa2? (see page 394) | <value> ::= integer or <string>  
<string> ::= "0xnn" n ::= (0,..,9 | A,..,F) |
| :TRIGger:IIC[:SOURce]:CLOCk <source> (see page 395) | :TRIGger:IIC[:SOURce]:CLOCk? (see page 395) | <source> ::= (CHANnel<n> | EXTernal) for DSO models  
<string> ::= (CHANnel<n> | DIGital0,...,DIGital15 ) for MSO models  
<n> ::= 1-2 or 1-4 in NRI format |
| :TRIGger:IIC[:SOURce]:DATa <source> (see page 396) | :TRIGger:IIC[:SOURce]:DATa? (see page 396) | <source> ::= (CHANnel<n> | EXTernal) for DSO models  
<string> ::= (CHANnel<n> | DIGital0,...,DIGital15 ) for MSO models  
<n> ::= 1-2 or 1-4 in NRI format |
| :TRIGger:IIC:TRIGger:QUAlifier <value> (see page 397) | :TRIGger:IIC:TRIGger:QUAlifier? (see page 397) | <value> ::= (EQUal | NOTequal | LESSthan | GREaterthan) |
| :TRIGger:IIC:TRIGger[:TYPe] <type> (see page 398) | :TRIGger:IIC:TRIGger[:TYPe]? (see page 398) | <type> ::= (START | STOP | READ7 | READEeprom | WRITe7 | WRITe10 | NACKnowledge | ANACKnowledge | R7Data2 | W7Data2 | REStart) |
392 Agilent 6000 Series Oscilloscopes Programmer’s Reference

3 Commands by Subsystem

:TRIGger:IIC:PATTern:ADDRess

(see page 564)

Command Syntax

:TRIGger:IIC:PATTern:ADDRess <value>

<value> ::= integer or <string>

<string> ::= "0xnn" where n ::= {0,...,9 | A,...,F}

The :TRIGger:IIC:PATTern:ADDRess command sets the address for IIC data. The address can range from 0x00 to 0x7F (7-bit) or 0x3FF (10-bit) hexadecimal. Use the don't care address (-1 or 0xFFFFFFFF) to ignore the address value.

Query Syntax

:TRIGger:IIC:PATTern:ADDRess?

The :TRIGger:IIC:PATTern:ADDRess? query returns the current address for IIC data.

Return Format

<value><NL>

<value> ::= integer

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:IIC:PATTern:DATA" on page 393
- ":TRIGger:IIC:PATTern:DATa2" on page 394
- "::TRIGger:IIC:TRIGger[:TYPE]" on page 398
**:TRIGger:IIC:PATTern:DATA**

(see page 564)

**Command Syntax**

:TRIGger:IIC:PATTern:DATA <value>

<value> ::= integer or <string>

<string> ::= "0xnn" where n ::= \{0,..,9 | A,...,F\}

The :TRIGger:IIC:PATTern:DATA command sets IIC data. The data value can range from 0x00 to 0x0FF (hexadecimal). Use the don't care data pattern (-1 or 0xFFFFFFFF) to ignore the data value.

**Query Syntax**

:TRIGger:IIC:PATTern:DATA?

The :TRIGger:IIC:PATTern:DATA? query returns the current pattern for IIC data.

**Return Format**

<value><NL>

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ".:TRIGger:IIC:PATTern:ADDRess" on page 392
- ".:TRIGger:IIC:PATTern:DATA2" on page 394
- ".:TRIGger:IIC:TRIGger[:TYPE]" on page 398
394 Agilent 6000 Series Oscilloscopes Programmer’s Reference

3 Commands by Subsystem

:TRIGger:IIC:PATTern:DATa2

(see page 564)

Command Syntax :TRIGger:IIC:PATTern:DATa2 <value>

<value> ::= integer or <string>

<string> ::= "0xnn" where n ::= {0,..,9 | A,..,F}

The :TRIGger:IIC:PATTern:DATa2 command sets IIC data 2. The data value can range from 0x00 to 0x0FF (hexadecimal). Use the don't care data pattern (-1 or 0xFFFFFFFF) to ignore the data value.

Query Syntax :TRIGger:IIC:PATTern:DATa2?


Return Format <value><NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- "TRIGger:IIC:PATTern:ADDRes" on page 392
- "TRIGger:IIC:PATTern:DATA" on page 393
- "TRIGger:IIC:TRIGger[:TYPE]" on page 398
**:TRIGger:IIC:SOURce:CLOCk**

(see page 564)

**Command Syntax**

`:TRIGger:IIC:[SOURce:]CLOCk <source>`

<source> ::= (CHANnel<n> | EXTernal) for the DSO models

<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

The :TRIGger:IIC:[SOURce:]CLOCk command sets the source for the IIC serial clock (SCL).

**Query Syntax**

`:TRIGger:IIC:[SOURce:]CLOCk?`

The :TRIGger: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 332
- ":TRIGger:IIC:SOURce:DATA" on page 396
Commands by Subsystem

:TRIGger:IIC:SOURce:DATA

(see page 564)

Command Syntax
:TRIGger:IIC:[SOURce:]DATA <source>

<source> ::= {CHANnel<n> | EXTernal} for the DSO models
<source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :TRIGger:IIC:[SOURce:]DATA command sets the source for IIC serial data (SDA).

Query Syntax
:TRIGger:IIC:[SOURce:]DATA?

The :TRIGger:IIC:[SOURce:]DATA? query returns the current source for IIC serial data.

Return Format
<source><NL>

See Also
- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:IIC:SOURce:CLOCk" on page 395
**:TRIGger:**IIC:**TRIGger:**QUALifier

(see page 564)

Command Syntax

```
:TRIGger:IIC:TRIGger:QUALifier <value>
```

\(<value> ::= \{EQUal \mid NOTequal \mid LESSthan \mid GREaterthan\}\)

The :TRIGger:IIC:TRIGger:QUALifier command sets the IIC data qualifier when TRIGger:IIC:TRIGger[:TYPE] is set to READEeprom.

Query Syntax

```
:TRIGger:IIC:TRIGger:QUALifier?
```

The :TRIGger:IIC:TRIGger:QUALifier? query returns the current IIC data qualifier value.

Return Format

```
<value><NL>
```

\(<value> ::= \{EQUal \mid NOTequal \mid LESSthan \mid GREaterthan\}\)

See Also

- "Introduction to :TRIGger Commands" on page 332
- "**:TRIGger:MODE**" on page 338
- "**:TRIGger:IIC:**TRIGger[:TYPE]" on page 398
:TRIGGER:IIC:TRIGGER[:TYPE]

Command Syntax

:TRIGGER:IIC:TRIGGER[:TYPE] <value>

<value> ::= {START | STOP | READ7 | READEeprom | WRITE7 | WRITe10 |
NACKnowledge | ANACKnowledge | R7Data2 | W7Data2 | RESTart}

The :TRIGGER:IIC:TRIGGER[:TYPE] command sets the IIC trigger type:

- **START** — Start condition.
- **STOP** — Stop condition.
- **READ7** — 7-bit address frame containing (Start:Address7:Read:Ack:Data). The value READ is also accepted for READ7.
- **R7Data2** — 7-bit address frame containing (Start:Address7:Read:Ack:Data:Ack:Data2).
- **READEeprom** — EEPROM data read.
- **WRITE7** — 7-bit address frame containing (Start:Address7:Write:Ack:Data). The value WRITe is also accepted for WRITe7.
- **W7Data2** — 7-bit address frame containing (Start:Address7:Write:Ack:Data:Ack:Data2).
- **WRITe10** — 10-bit address frame containing (Start:Address byte1:Write:Ack:Address byte 2:Data).
- **NACKnowledge** — Missing acknowledge.
- **ANACKnowledge** — Address with no acknowledge.
- **RESTart** — Another start condition occurs before a stop condition.

**NOTE**
The short form of READ7 (READ7), READEeprom (READE), WRITE7 (WRITe7), and WRITE10 (WRITe10) do not follow the defined Long Form to Short Form Truncation Rules (see page 566).

Query Syntax

:TRIGGER:IIC:TRIGGER[:TYPE]?

The :TRIGGER:IIC:TRIGGER[:TYPE]? query returns the current IIC trigger type value.

Return Format

<value><NL>

<value> ::= {START | STOP | READ7 | READE | WRITe7 | WRITe10 | NACK | ANAC |
R7D2 | W7D2 | REST}

See Also

- "Introduction to TRIGGER Commands" on page 332
- "TRIGGER:MODE" on page 338
- ":TRIGger:IIC:PATTern:ADDRess" on page 392
- ":TRIGger:IIC:PATTern:DATA" on page 393
- ":TRIGger:IIC:PATTern:DATa2" on page 394
- ":TRIGger:IIC:TRIGger:QUALifier*" on page 397
- "Long Form to Short Form Truncation Rules*" on page 566
### :TRIGger:LIN Commands

#### Table 70 :TRIGger:LIN Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:LIN:ID &lt;value&gt;</td>
<td>:TRIGger:LIN:ID? (see page 401)</td>
<td>&lt;value&gt; ::= 7-bit integer in decimal, &lt;nondecimal&gt;, or &lt;string&gt; from 0-63 or 0x00-0x3f (with Option AMS)</td>
</tr>
<tr>
<td>(see page 401)</td>
<td></td>
<td>&lt;nondecimal&gt; ::= #Hnn where n ::= (0,..,9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;nondecimal&gt; ::= #Bnn...n where n ::= (0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;string&gt; ::= &quot;0xnn&quot; where n ::= (0,..,9</td>
</tr>
<tr>
<td>:TRIGger:LIN:SAMPLEpoint &lt;value&gt;</td>
<td>:TRIGger:LIN:SAMPLEpoint? (see page 402)</td>
<td>&lt;value&gt; ::= 60</td>
</tr>
<tr>
<td>(see page 402)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:SIGNAL:BAUDrate &lt;baudrate&gt;</td>
<td>:TRIGger:LIN:SIGNAL:BAUDrate? (see page 403)</td>
<td>&lt;baudrate&gt; ::= (2400</td>
</tr>
<tr>
<td>(see page 403)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:SOURce &lt;source&gt;</td>
<td>:TRIGger:LIN:SOURce? (see page 404)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>(see page 404)</td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:LIN:STANDard &lt;std&gt;</td>
<td>:TRIGger:LIN:STANDard? (see page 405)</td>
<td>&lt;std&gt; ::= (LIN13</td>
</tr>
<tr>
<td>(see page 405)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:SYNCbreak &lt;value&gt;</td>
<td>:TRIGger:LIN:SYNCbreak? (see page 406)</td>
<td>&lt;value&gt; ::= integer = (11</td>
</tr>
<tr>
<td>(see page 406)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:TRIGger &lt;condition&gt;</td>
<td>:TRIGger:LIN:TRIGger? (see page 407)</td>
<td>&lt;condition&gt; ::= (SYNCbreak) (without Option AMS)</td>
</tr>
<tr>
<td>(see page 407)</td>
<td></td>
<td>&lt;condition&gt; ::= (SYNCbreak</td>
</tr>
</tbody>
</table>
**:TRIGger:LIN:ID**

(see page 564)

**Command Syntax**

**:TRIGger:LIN:ID <value>**

<value> ::= 7-bit integer in decimal, <nondecimal>, or <string>
from 0-63 or 0x00-0x3f

<nondecimal> ::= #Hnn where n ::= {0,..,9 \mid A,..,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

<string> ::= "0xnn" where n ::= {0,..,9 \mid A,..,F} for hexadecimal

The :TRIGger:LIN:ID command defines the LIN identifier searched for in each CAN message when the LIN trigger mode is set to frame ID.

**NOTE**

This command is only valid when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

**Query Syntax**

**:TRIGger:LIN:ID?**

The :TRIGger:LIN:ID? query returns the current LIN identifier setting.

**Return Format**

<value><NL>

<value> ::= integer in decimal

**Errors**

- "-241, Hardware missing" on page 535

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:TRIGger" on page 407
- ":TRIGger:LIN:SIGNAL:DEFinition" on page 530
- ":TRIGger:LIN:SOURce" on page 404
**:TRIGger:LIN:SAMPlepoint**

**Command Syntax**

\[ \text{:TRIGger:LIN:SAMPlepoint } <\text{value}> \]

\[ <\text{value}> ::= \{ 60 \mid 62.5 \mid 68 \mid 70 \mid 75 \mid 80 \mid 87.5 \} \text{ in NR3 format} \]

The :TRIGger:LIN:SAMPlepoint command sets the point during the bit time where the bit level is sampled to determine whether the bit is dominant or recessive. The sample point represents the percentage of time between the beginning of the bit time to the end of the bit time.

**NOTE**

The sample point values are not limited by the baud rate.

**Query Syntax**

\[ \text{:TRIGger:LIN:SAMPlepoint?} \]

The :TRIGger:LIN:SAMPlepoint? query returns the current LIN sample point setting.

**Return Format**

\[ <\text{value}> ::= \{ 60 \mid 62.5 \mid 68 \mid 70 \mid 75 \mid 80 \mid 87.5 \} \text{ in NR3 format} \]

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:TRIGger" on page 407
:TRIGger:LIN:SIGNal:BAUDrate

(see page 564)

Command Syntax

:TRIGger:LIN:SIGNal:BAUDrate <baudrate>

<baudrate> ::= integer in NR1 format

<baudrate> ::= (2400 | 9600 | 19200)

The :TRIGger:LIN:SIGNal:BAUDrate command sets the standard baud rate of the LIN signal at 2400 b/s, 9600 b/s, or 19200 b/s. If a non-standard baud rate is sent, the baud rate will be set to the next highest standard rate.

Query Syntax

:TRIGger:LIN:SIGNal:BAUDrate?

The :TRIGger:LIN:SIGNal:BAUDrate? query returns the current LIN baud rate setting.

Return Format

<baudrate><NL>

<baudrate> ::= integer = (2400 | 9600 | 19200)

See Also

- "Introduction to :TRIGger Commands" on page 332
- "TRIGger:MODE" on page 338
- "TRIGger:LIN:TRIGger" on page 407
- "TRIGger:LIN:SIGNal:DEFinition" on page 530
- "TRIGger:LIN:SOURce" on page 404
**:TRIGger:LIN:SOURce**

(see page 564)

**Command Syntax**

`:TRIGger:LIN:SOURce <source>`

- `<source>` ::= (CHANnel<`n`> | EXTernal) for the DSO models
- `<source>` ::= (CHANnel<`n`> | DIGital0,...,DIGital15) for the MSO models
- `<n>` ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
- `<n>` ::= (1 | 2) for the two channel oscilloscope models

The :TRIGger:LIN:SOURce command sets the source for the LIN signal.

**Query Syntax**

`:TRIGger:LIN:SOURce?`

The :TRIGger:LIN:SOURce? query returns the current source for the LIN signal.

**Return Format**

`<source><NL>`

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:TRIGger" on page 407
- ":TRIGger:LIN:SIGNal:DEFinition" on page 530
**:TRIGger:LIN:STANdard**

(see page 564)

**Command Syntax**

`:TRIGger:LIN:STANdard <std>`

<std> ::= (LIN13 | LIN20)

The `:TRIGger:LIN:STANdard` command sets the LIN standard in effect for triggering and decoding to be LIN1.3 or LIN2.0.

**Query Syntax**

`:TRIGger:LIN:STANdard?`

The `:TRIGger:LIN:STANdard?` query returns the current LIN standard setting.

**Return Format**

`<std><NL>`

<std> ::= (LIN13 | LIN20)

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":.TRIGger:MODE" on page 338
- ":.TRIGger:LIN:SIGNal:DEFinition" on page 530
- ":.TRIGger:LIN:SOURce" on page 404
**:TRIGger:LIN:SYNCbreak**

(see page 564)

**Command Syntax**

`:TRIGger:LIN:SYNCbreak <value>`

<value> ::= integer = {11  |  12  |  13}

The :TRIGger: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**

`:TRIGger:LIN:SYNCbreak?`

The :TRIGger:LIN:STANdard? query returns the current LIN sync break setting.

**Return Format**

<value><NL>

<value> ::= {11  |  12  |  13}

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:SIGNal:DEFinition" on page 530
- ":TRIGger:LIN:SOURce" on page 404
:TRIGger:LIN:TRIGger

(see page 564)

Command Syntax

:TRIGger:LIN:TRIGger <condition>

<condition> ::= (SYNCbreak | ID)

The :TRIGger:LIN:TRIGger command sets the LIN trigger on condition to be Sync Break (SYNCbreak) or Frame Id (ID).

NOTE

The ID option is available when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

Query Syntax

:TRIGger:LIN:TRIGger?

The :TRIGger:LIN:TRIGger? query returns the current LIN trigger value.

Return Format

<condition><NL>

<condition> ::= (SYNC | ID)

Errors

- "-241, Hardware missing" on page 535

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:SIGNal:DEFinition" on page 530
- ":TRIGger:LIN:SOURce" on page 404
### :TRIGger:SEQUence Commands

#### Table 71 :TRIGger:SEQUence Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:SEQUence:COUNt &lt;count&gt;</td>
<td>:TRIGger:SEQUence:COUNt?</td>
<td>&lt;count&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:EDGE{1</td>
<td>2} &lt;source&gt;, &lt;slope&gt;</td>
<td>:TRIGger:SEQUence:EDGE{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;slope&gt; ::= (POSitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;n&gt; ::= 1-2 or 1-4 in NR1 format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;return_value&gt; ::= query returns &quot;NONE&quot; if edge source is disabled</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:FIND &lt;value&gt;</td>
<td>:TRIGger:SEQUence:FIND?</td>
<td>&lt;value&gt; ::= (PATTern1,ENTered</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:PA TTern{1</td>
<td>2} &lt;value&gt;, &lt;mask&gt;</td>
<td>:TRIGger:SEQUence:PA TTern{1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;mask&gt; ::= integer or &lt;string&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;string&gt; ::= &quot;0xnnnnnn&quot; n ::= (0,...9</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:RESet &lt;value&gt;</td>
<td>:TRIGger:SEQUence:RESet?</td>
<td>&lt;value&gt; ::= (NONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Values used in find and trigger stages not available. EDGE2 not available if EDGE2,COUNt used in trigger stage.</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:TIMer &lt;time_value&gt;</td>
<td>:TRIGger:SEQUence:TIMer?</td>
<td>&lt;time_value&gt; ::= time from 100 ns to 10 seconds in NR3 format</td>
</tr>
<tr>
<td>:TRIGger:SEQUence:TRIGger &lt;value&gt;</td>
<td>:TRIGger:SEQUence:TRIGger?</td>
<td>&lt;value&gt; ::= (PA TTern2,ENTered</td>
</tr>
</tbody>
</table>
:TRIGger:SEQUence:COUNt

Command Syntax

:TRIGger:SEQUence:COUNt <count>

<count> ::= integer in NR1 format

The :TRIGger:SEQUence:COUNt command sets the sequencer edge counter resource. The edge counter is used in the trigger stage to determine the number of edges that must be found before the sequencer generates a trigger.

Query Syntax

:TRIGger:SEQUence:COUNt?

The :TRIGger:SEQUence:COUNt? query returns the current sequencer edge counter setting.

Return Format

<count><NL>

<count> ::= integer in NR1 format

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQUence:TRIGger" on page 415
- ":TRIGger:SEQUence:EDGE" on page 410
3 Commands by Subsystem

:TRIGger:SEQUence:EDGE

(see page 564)

Command Syntax

:TRIGger:SEQUence:EDGE{(1 | 2) <source>, <slope>}

<source> ::= (CHANnel<n> | EXTernal) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,..,DIGital15) for the MSO models
<slope> ::= (POSitive | NEGative)

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :TRIGger:SEQUence:EDGE<n> command defines the specified sequencer edge resource according to the specified <source> and <slope>. To disable an edge resource, set its <source> to NONE. In this case, <slope> has no meaning.

Query Syntax

:TRIGger:SEQUence:EDGE{(1 | 2)?

The :TRIGger:SEQUence:EDGE<n>? query returns the specified sequencer edge resource setting. If the edge resource is disabled, the returned <source> value is NONE. In this case, the <slope> is undefined.

Return Format

<source>, <slope><NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQUence:FIND" on page 411
- ":TRIGger:SEQUence:TRIGger" on page 415
- ":TRIGger:SEQUence:RESet" on page 413
- ":TRIGger:SEQUence:COUNT" on page 409
**:TRIGger:SEQUence:FIND**

(see page 564)

**Command Syntax**

```
:TRIGger:SEQUence:FIND <value>
```

`<value>` ::= (PATTern1,ENTERed | PATTern1,EXITed | EDGE1 | PATTern1,AND,EDGE1)

The :TRIGger:SEQUence:FIND command specifies the find stage of a sequence trigger. This command accepts three program data parameters; you can use NONE to fill out the parameter list (for example,"EDGE1,NONE,NONE").

PATTern1 is specified with the":TRIGger:SEQUence:PATTern command. EDGE1 is specified with the :TRIGger:SEQUence:EDGE command.

**Query Syntax**

```
:TRIGger:SEQUence:FIND?
```

The :TRIGger:SEQUence:FIND? query returns the find stage specification for a sequence trigger. NONE is returned for unused parameters.

**Return Format**

```
<find_value><NL>
```

`<find_value>` ::= (PATT1,ENT,NONE | PATT1,EXIT,NONE | EDGE1,NONE,NONE | PATT1,AND,EDGE1)

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQUence:PATTern" on page 412
- ":TRIGger:SEQUence:EDGE" on page 410
- ":TRIGger:SEQUence:TRIGger" on page 415
- ":TRIGger:SEQUence:RESet" on page 413
Commands by Subsystem

:TRIGger:SEQuence:PATTern

(see page 564)

Command Syntax

:TRIGger:SEQuence:PATTern{1 | 2} <value>,<mask>

<value> ::= integer or <string>

<mask> ::= integer or <string>

<string> ::= "0xnnnnnn" where n ::= {0,..,9 | A,..,F}

The :TRIGger:SEQuence:PATTern<n> command defines the specified sequence pattern resource according to the value and the mask. For both <value> and <mask>, each bit corresponds to a possible trigger channel. The bit assignments vary by instrument:

<table>
<thead>
<tr>
<th>Oscilloscope Models</th>
<th>Value and Mask Bit Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 through 19 - analog channels 1 through 4.</td>
</tr>
<tr>
<td>2 analog + 16 digital channels (mixed-signal)</td>
<td>Bits 0 through 15 - digital channels 0 through 15. Bits 16 and 17 - analog channels 1 and 2.</td>
</tr>
<tr>
<td>4 analog channels only</td>
<td>Bits 0 through 3 - analog channels 1 through 4. Bit 4 - external trigger.</td>
</tr>
<tr>
<td>2 analog channels only</td>
<td>Bits 0 and 1 - analog channels 1 and 2. Bit 4 - external trigger.</td>
</tr>
</tbody>
</table>

Set a <value> bit to "0" to set the pattern for the corresponding channel to low. Set a <value> bit to "1" to set the pattern to high.

Set a <mask> bit to "0" to ignore the data for the corresponding channel. Only channels with a "1" set on the appropriate mask bit are used.

Query Syntax

:TRIGger:SEQuence:PATTern{1 | 2}??

The :TRIGger:SEQuence:PATTern<n> query returns the current settings of the specified pattern resource.

Return Format

<value>, <mask><NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQuence:FIND" on page 411
- ":TRIGger:SEQuence:TRIGger" on page 415
- ":TRIGger:SEQuence:RESet" on page 413
**:TRIgger:SEQUence:RESet**

(see page 564)

**Command Syntax**

```
:TRIgger:SEQUence:RESet <value>
```

```
<value> ::= (NONE | PATTern1,ENTERed | PATTern1,EXITed | EDGE1
| PATTern1,AND,EDGE1 | PATTern2,ENTERed | PATTern2,EXITed
| EDGE2 | TIMer)
```

Values used in find and trigger stages are not available. EDGE2 is not available if EDGE2,COUNT is used in trigger stage.

The :TRIgger:SEQUence:RESet command specifies the reset stage of a sequence trigger. In multi-level trigger specifications, you may find a pattern, then search for another in sequence, but reset the entire search to the beginning if another condition occurs. This command accepts three program data parameters; you can use NONE to fill out the parameter list (for example, "EDGE1,NONE,NONE").

PATTern1 and PATTern2 are specified with the :TRIgger:SEQUence:PATTern command. EDGE1 and EDGE2 are specified with the :TRIgger:SEQUence:EDGE command. TIMer is specified with the :TRIgger:SEQUence:TIMer command.

**Query Syntax**

```
:TRIgger:SEQUence:RESet?
```

The :TRIgger:SEQUence:RESet? query returns the reset stage specification for a sequence trigger. NONE is returned for unused parameters.

**Return Format**

```
<reset_value><NL>
```

```
<reset_value> ::= (NONE,NONE,NONE | PATT1,ENT,NONE | PATT1,EXIT,NONE
| EDGE1,NONE,NONE | PATT1,AND,EDGE1 | PATT2,ENTER,NONE
| PATT2,EXIT,NONE | EDGE2,NONE,NONE | TIM,NONE,NONE)
```

**See Also**

- "Introduction to TRIgger Commands" on page 332
- ":TRIgger:SEQUence:PATTern" on page 412
- ":TRIgger:SEQUence:EDGE" on page 410
- ":TRIgger:SEQUence:TIMer" on page 414
- ":TRIgger:SEQUence:FIND" on page 411
- ":TRIgger:SEQUence:TRIgger" on page 415
414 Agilent 6000 Series Oscilloscopes Programmer’s Reference

3 Commands by Subsystem

:TRIGger:SEQuence:TIMer

(see page 564)

Command Syntax

:TRIGger:SEQuence:TIMer <time_value>

<time_value> ::= time in seconds in NR1 format

The :TRIGger:SEQuence:TIMer command sets the sequencer timer resource in seconds from 100 ns to 10 s. The timer is used in the reset stage to determine how long to wait for the trigger to occur before restarting.

Query Syntax

:TRIGger:SEQuence:TIMer?

The :TRIGger:SEQuence:TIMer? query returns current sequencer timer setting.

Return Format

<time_value><NL>

<time_value> ::= time in seconds in NR1 format

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQuence:RESet" on page 413
:TRIGger:SEQuence:TRIGger

\[(\text{see page 564})\]

Command Syntax

\[
:TRIGger:SEQuence:TRIGger \ <value>
\]

\[
<value> ::= (\text{PATTern2,ENTERed | PATTern2,EXITed | EDGE2} \\
| \text{PATTern2,AND,EDGE2 | EDGE2,COUNT | EDGE2,COUNT,NREFind})
\]

The :TRIGger:SEQuence:TRIGger command specifies the trigger stage of a sequence trigger. The sequence commands set various search terms. After all of these are found in sequence, the trigger condition itself is searched for. This command accepts three program data parameters; you can use NONE to fill out the parameter list (for example, "EDGE2,NONE,NONE").

PATTern2 is specified with the :TRIGger:SEQuence:PATTern command.
EDGE2 is specified with the :TRIGger:SEQuence:EDGE command. COUNT is specified with the :TRIGger:SEQuence:COUNT command.

Query Syntax

\[
:TRIGger:SEQuence:TRIGger?\]

The :TRIGger:SEQuence:TRIGger? query returns the trigger stage specification for a sequence trigger. NONE is returned for unused parameters.

Return Format

\[
<trigger_value><NL>
\]

\[
<trigger_value> ::= \{\text{PATT2,ENT,NONE | PATT2,EXIT,NONE} \\
| \text{EDGE2,NONE,NONE | PATT2,AND,EDGE2} \\
| \text{EDGE2,COUN,NONE | EDGE2,COUN,NREF}\}
\]

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SEQuence:PATTern" on page 412
- ":TRIGger:SEQuence:EDGE" on page 410
- ":TRIGger:SEQuence:COUNT" on page 409
- ":TRIGger:SEQuence:FIND" on page 411
- ":TRIGger:SEQuence:RESet" on page 413
- ":TRIGger:SEQuence:RESet" on page 413
### :TRIGger:SPI Commands

**Table 72 :TRIGger:SPI Commands Summary**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:SPI:CLOCK:SLOPe &lt;slope&gt; (see page 417)</td>
<td>:TRIGger:SPI:CLOCK:SLOPe? (see page 417)</td>
<td>&lt;slope&gt; ::= (NEGative</td>
</tr>
<tr>
<td>:TRIGger:SPI:CLOCK:TIMEout &lt;time_value&gt; (see page 418)</td>
<td>:TRIGger:SPI:CLOCK:TIMEout? (see page 418)</td>
<td>&lt;time_value&gt; ::= time in seconds in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:SPI:FRAMing &lt;value&gt; (see page 419)</td>
<td>:TRIGger:SPI:FRAMing? (see page 419)</td>
<td>&lt;value&gt; ::= (CHIPselect</td>
</tr>
<tr>
<td>:TRIGger:SPI:PATTern:DATA &lt;value&gt;, &lt;mask&gt; (see page 420)</td>
<td>:TRIGger:SPI:PATTern:DATA? (see page 420)</td>
<td>&lt;value&gt; ::= integer or &lt;string&gt; &lt;mask&gt; ::= integer or &lt;string&gt; &lt;string&gt; ::= &quot;0xnnnnnn&quot; where n ::= {0,..,9</td>
</tr>
<tr>
<td>:TRIGger:SPI:PATTern:WIDTH &lt;width&gt; (see page 421)</td>
<td>:TRIGger:SPI:PATTern:WIDTH? (see page 421)</td>
<td>&lt;width&gt; ::= integer from 4 to 32 in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:CLOCK &lt;source&gt; (see page 422)</td>
<td>:TRIGger:SPI:SOURce:CLOCK? (see page 422)</td>
<td>&lt;value&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:DATA &lt;source&gt; (see page 423)</td>
<td>:TRIGger:SPI:SOURce:DATA? (see page 423)</td>
<td>&lt;value&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGger:SPI:SOURce:FRAme &lt;source&gt; (see page 424)</td>
<td>:TRIGger:SPI:SOURce:FRAme? (see page 424)</td>
<td>&lt;value&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
</tbody>
</table>
:TRIGger:SPI:CLOCk:SLOPe

(see page 564)

Command Syntax

:TRIGger:SPI:CLOCk:SLOPe <slope>

<slope> ::= {NEGative | POSitive}

The :TRIGger: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

:TRIGger:SPI:CLOCk:SLOPe?

The :TRIGger: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 332
- ".:TRIGger:SPI:CLOCk:TIMEout" on page 418
- ".:TRIGger:SPI:SOURce:CLOCk" on page 422
3 Commands by Subsystem

:TRIGger:SPI:CLOck:TIMeout

(see page 564)

Command Syntax

:TRIGger:SPI:CLOck:TIMeout <time_value>

<time_value> ::= time in seconds in NR1 format

The :TRIGger:SPI:CLOck:TIMeout command sets the SPI signal clock timeout resource in seconds from 500 ns to 10 s when the :TRIGger:SPI:FRAMing command is set to TIMeout. The timer is used to frame a signal by a clock timeout.

Query Syntax

:TRIGger:SPI:CLOck:TIMeout?


Return Format

<time value><NL>

<time_value> ::= time in seconds in NR1 format

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SPI:CLOck:SLOPe" on page 417
- ":TRIGger:SPI:SOURce:CLOck" on page 422
- ":TRIGger:SPI:FRAMing" on page 419
**:TRIGger:SPI:FRAMing**

(see page 564)

**Command Syntax**

`:TRIGger:SPI:FRAMing <value>`

<value> ::= (CHIPselect | NOTChipselect | TIMEout)

The :TRIGger:SPI:FRAMing command sets the SPI trigger framing value. If TIMEout is selected, the timeout value is set by the :TRIGger:SPI:CLOCk:TIMeout command.

**Query Syntax**

`:TRIGger:SPI:FRAMing?`

The :TRIGger:SPI:FRAMing? query returns the current SPI framing value.

**Return Format**

<value><NL>

<value> ::= (CHIPselect | NOTChipselect | TIMEout)

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:SPI:CLOCk:TIMeout" on page 418
- ":TRIGger:SPI:SOURce:FRAMe" on page 424
3 Commands by Subsystem

:TRIGger:SPI:PATTern:DATA

(see page 564)

Command Syntax

:TRIGger:SPI:PATTern:DATA <value>,<mask>

<value> ::= integer or <string>

<mask> ::= integer or <string>

<string> ::= "0xnnnnnn" where n ::= {0,...,9 | A,...,F}

The :TRIGger:SPI:PATTern:DATA command defines the SPI data pattern resource according to the value and the mask. This pattern, along with the data width, control the data pattern searched for in the data stream.

Set a <value> bit to "0" to set the corresponding bit in the data pattern to low. Set a <value> bit to "1" to set the bit to high.

Set a <mask> bit to "0" to ignore that bit in the data stream. Only bits with a "1" set on the mask are used.

Query Syntax

:TRIGger:SPI:PATTern:DATA?


Return Format

<value>, <mask><NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:SPI:PATTern:WIDTh" on page 421
- ":TRIGger:SPI:SOURce:DATA" on page 423
Commands by Subsystem

:TRIgger:SPI:PARTtern:WIDTh

(see page 564)

Command Syntax

:TRIgger:SPI:PARTtern:WIDTh <width>

<width> ::= integer from 4 to 32 in NR1 format

The :TRIgger:SPI:PARTtern:WIDTh command sets the width of the SPI data pattern anywhere from 4 bits to 32 bits.

Query Syntax

:TRIgger:SPI:PARTtern:WIDTh?

The :TRIgger:SPI:PARTtern:WIDTh? query returns the current SPI data pattern width setting.

Return Format

<width><NL>

<width> ::= integer from 4 to 32 in NR1 format

See Also

- "Introduction to :TRIgger Commands" on page 332
- "TRIgger:SPI:PARTtern:DATA" on page 420
- "TRIgger:SPI:SOURce:DATA" on page 423
3 Commands by Subsystem

:TRIGger:SPI:SOURce:CLOCk

(see page 564)

Command Syntax

:TRIGger:SPI:SOURce:CLOCk <source>

<source> ::= (CHANnel<n> | EXTernal) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models
<n> ::= {1, 2, 3, 4} for the four channel oscilloscope models
<n> ::= {1, 2} for the two channel oscilloscope models

The :TRIGger:SPI:SOURce:CLOCk command sets the source for the SPI serial clock.

Query Syntax

:TRIGger:SPI:SOURce:CLOCk?

The :TRIGger: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 332
- ":.TRIGger:SPI:CLOCk:SLOPe" on page 417
- ":.TRIGger:SPI:CLOCk:TIMEout" on page 418
- ":.TRIGger:SPI:SOURce:FRAMe" on page 424
- ":.TRIGger:SPI:SOURce:DATA" on page 423
:TRIGGER:SPI:SOURce:DATA

(see page 564)

Command Syntax
:TRIGGER:SPI:SOURce:DATA <source>

<source> ::= (CHANnel<n> | EXternal) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :TRIGGER:SPI:SOURce:DATA command sets the source for the SPI serial data.

Query Syntax
:TRIGGER:SPI:SOURce:DATA?

The :TRIGGER:SPI:SOURce:DATA? query returns the current source for the SPI serial data.

Return Format
<source><NL>

See Also
- "Introduction to :TRIGGER Commands" on page 332
- "TRIGGER:SPI:SOURce:CLOCK" on page 422
- "TRIGGER:SPI:SOURce:FRAME" on page 424
- "TRIGGER:SPI:PATTERN:DATA" on page 420
- "TRIGGER:SPI:PATTERN:WIDTH" on page 421
3 Commands by Subsystem

**:TRIGger:SPI:SOURce:FRAMe**

(see page 564)

**Command Syntax**

**:TRIGger:SPI:SOURce:FRAMe <source>**

<source> ::= (CHANnel<n> | EXTernal) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models
<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models

The **:TRIGger:SPI:SOURce:FRAMe** command sets the frame source when **:TRIGger:SPI:FRAMing** is set to CHIPselect or NOTChipselect.

**Query Syntax**

**:TRIGger:SPI:SOURce:FRAMe?**

The **:TRIGger: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 332
- ":TRIGger:SPI:SOURce:CLOCk" on page 422
- ":TRIGger:SPI:SOURce:DATA" on page 423
- ":TRIGger:SPI:FRAMing" on page 419
**:TRIGger:TV Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGger:TV:LINE &lt;line number&gt; (see page 426)</td>
<td>:TRIGger:TV:LINE? (see page 426)</td>
<td>&lt;line number&gt; ::= integer in NR1 format</td>
</tr>
<tr>
<td>:TRIGger:TV:MODE &lt;tv mode&gt; (see page 427)</td>
<td>:TRIGger:TV:MODE? (see page 427)</td>
<td>&lt;tv mode&gt; ::= {FIELD1</td>
</tr>
<tr>
<td>:TRIGger:TV:POLarity &lt;polarity&gt; (see page 428)</td>
<td>:TRIGger:TV:POLarity? (see page 428)</td>
<td>&lt;polarity&gt; ::= {POSitive</td>
</tr>
<tr>
<td>:TRIGger:TV:SOURce &lt;source&gt; (see page 429)</td>
<td>:TRIGger:TV:SOURce? (see page 429)</td>
<td>&lt;source&gt; ::= (CHANNEL&lt;n&gt;)</td>
</tr>
<tr>
<td>:TRIGger:TV:STANdard &lt;standard&gt; (see page 430)</td>
<td>:TRIGger:TV:STANdard? (see page 430)</td>
<td>&lt;standard&gt; ::= {GENeric</td>
</tr>
</tbody>
</table>
3  Commands by Subsystem

:TRIGger:TV:LINE

(see page 564)

Command Syntax

[:TRIGger:TV:LINE <line_number>]

<line_number> ::= integer in NR1 format

The :TRIGger:TV:LINE command allows triggering on a specific line of video. The line number limits vary with the standard and mode, as shown in the following table.

Table 74  TV Trigger Line Number Limits

<table>
<thead>
<tr>
<th>TV Standard</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LINE</td>
</tr>
<tr>
<td>NTSC</td>
<td>1 to 263</td>
</tr>
<tr>
<td>PAL</td>
<td>1 to 313</td>
</tr>
<tr>
<td>PAL-M</td>
<td>1 to 263</td>
</tr>
<tr>
<td>SECAM</td>
<td>1 to 313</td>
</tr>
<tr>
<td>GENERIC</td>
<td>1 to 1024</td>
</tr>
<tr>
<td>P480L60HZ</td>
<td>1 to 525</td>
</tr>
<tr>
<td>P720L60HZ</td>
<td>1 to 750</td>
</tr>
<tr>
<td>P1080L24HZ</td>
<td>1 to 1125</td>
</tr>
<tr>
<td>P1080L25HZ</td>
<td>1 to 1125</td>
</tr>
<tr>
<td>I1080L50HZ</td>
<td>1 to 1125</td>
</tr>
<tr>
<td>I1080L60HZ</td>
<td>1 to 1125</td>
</tr>
</tbody>
</table>

Query Syntax

[:TRIGger:TV:LINE?]

The :TRIGger:TV:LINE? query returns the current TV trigger line number setting.

Return Format

<line_number><NL>

<line_number> ::= integer in NR1 format

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:TV:STANdard" on page 430
- ":TRIGger:TV:MODE" on page 427
**Command Syntax**

:TRIGger:TV:MODE <mode>

<mode> ::= {FIELd1 | FIELd2 | AFIelds | ALINes | LINE | VERTical |
| LFIeld1 | LFIeld2 | LALTernate | LVERTical}

The :TRIGger:TV:MODE command selects the TV trigger mode and field. The LVERTical parameter is only available when :TRIGger:TV:STANdard is GENeric. The LALTernate parameter is not available when :TRIGger:TV:STANdard is GENeric.

Old forms for <mode> are accepted:

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Old Forms Accepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIELd1</td>
<td>F1</td>
</tr>
<tr>
<td>FIELd2</td>
<td>F2</td>
</tr>
<tr>
<td>AFIelds</td>
<td>ALLFields, ALLFLDS</td>
</tr>
<tr>
<td>ALINes</td>
<td>ALLLines</td>
</tr>
<tr>
<td>LFIeld1</td>
<td>LINEF1, LINEFIELD1</td>
</tr>
<tr>
<td>LFIeld2</td>
<td>LINEF2, LINEFIELD2</td>
</tr>
<tr>
<td>LALTernate</td>
<td>LINEAlt</td>
</tr>
<tr>
<td>LVERTical</td>
<td>LINEVert</td>
</tr>
</tbody>
</table>

**Query Syntax**

:TRIGger:TV:MODE?

The :TRIGger:TV:MODE? query returns the TV trigger mode.

**Return Format**

<value><NL>

<value> ::= {FIEL1 | FIEL2 | AFI | ALIN | LINE | VERT | LFI1 | LFI2 |
| LALT | LVERT}

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:TV:STANdard" on page 430
- ":TRIGger:MODE" on page 338
3 Commands by Subsystem

**:TRIGger:TV:POLarity**

(see page 564)

**Command Syntax**

:TRIGger:TV:POLarity <polarity>

<polarity> ::= {POSitive | NEGative}

The :TRIGger:TV:POLarity command sets the polarity for the TV trigger.

**Query Syntax**

:TRIGger:TV:POLarity?

The :TRIGger:TV:POLarity? query returns the TV trigger polarity.

**Return Format**

<polarity><NL>

<polarity> ::= {POS | NEG}

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:TV:SOURce" on page 429
:TRIGGER:TV:SOURce

Command Syntax
:TRIGGER:TV:SOURce <source>
<source> ::= {CHANnel<n>}
<n> ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
<n> ::= (1 | 2) for the two channel oscilloscope models

The :TRIGGER:TV:SOURce command selects the channel used to produce the trigger.

Query Syntax
:TRIGGER:TV:SOURce?

The :TRIGGER:TV:SOURce? query returns the current TV trigger source.

Return Format
<source><NL>
<source> ::= {CHAN<n>}

See Also
- "Introduction to :TRIGGER Commands" on page 332
- "TRIGGER:MODE" on page 338
- "TRIGGER:POLarity" on page 428

Example Code
- "Example Code" on page 369
3 Commands by Subsystem

:TRIGger:TV:STANdard

N (see page 564)

Command Syntax
:TRIGger:TV:STANdard <standard>

<standard> ::= {GENeric | NTSC | PALM | PAL | SECam |
| (P480L60HZ | P480) | (P720L60HZ | P720) |
| (P1080L24HZ | P1080) | P1080L25HZ |
| (I1080L50HZ | I1080) | I1080L60HZ}

The :TRIGger:TV:STANdard command selects the video standard. GENeric mode is non-interlaced.

Query Syntax
:TRIGger:TV:STANdard?

The :TRIGger:TV:STANdard? query returns the current TV trigger standard setting.

Return Format
<standard><NL>

<standard> ::= {GEN | NTSC | PALM | PAL | SEC | P480L60HZ | P760L60HZ |
P1080L24HZ | P1080L25HZ | I1080L50HZ | I1080L60HZ}
## :TRIGGER:USB Commands

### Table 75 :TRIGGER:USB Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:TRIGGER:USB:SOURce:DMI Nus &lt;source&gt; (see page 432)</td>
<td>:TRIGGER:USB:SOURce:DMI Nus? (see page 432)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGGER:USB:SOURce:DPLUS &lt;source&gt; (see page 433)</td>
<td>:TRIGGER:USB:SOURce:DPLUS? (see page 433)</td>
<td>&lt;source&gt; ::= (CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td>:TRIGGER:USB:SPEED &lt;value&gt; (see page 434)</td>
<td>:TRIGGER:USB:SPEED? (see page 434)</td>
<td>&lt;value&gt; ::= (LOW</td>
</tr>
<tr>
<td>:TRIGGER:USB:TRIGGER &lt;value&gt; (see page 435)</td>
<td>:TRIGGER:USB:TRIGGER? (see page 435)</td>
<td>&lt;value&gt; ::= (SOP</td>
</tr>
</tbody>
</table>
3 Commands by Subsystem

:TRIGger:USB:SOURce:DMINus

(see page 564)

**Command Syntax**

:TRIGger:USB:SOURce:DMINus <source>

<source> ::= (CHANnel<n> | EXternal) for the DSO models
<source> ::= (CHANnel<n> | DIGital0,...,DIGital15) for the MSO models

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The :TRIGger:USB:SOURce:DMINus command sets the source for the USB D- signal.

**Query Syntax**

:TRIGger:USB:SOURce:DMINus?

The :TRIGger:USB:SOURce:DMINus? query returns the current source for the USB D- signal.

**Return Format**

<source><NL>

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:USB:SOURce:DPLus" on page 433
- ":TRIGger:USB:TRIGger" on page 435
**:TRIGger:USB:SOURce:DPLus**

(see page 564)

**Command Syntax**

```
:TRIGger:USB:SOURce:DPLus <source>
```

- `<source>` ::= (CHANNEL<n> | EXternal) for the DSO models
- `<source>` ::= (CHANNEL<n> | DIGital0,...,DIGital15) for the MSO models
- `<n>` ::= (1 | 2 | 3 | 4) for the four channel oscilloscope models
- `<n>` ::= (1 | 2) for the two channel oscilloscope models

The :TRIGger:USB:SOURce:DPLus command sets the source for the USB D+ signal.

**Query Syntax**

```
:TRIGger:USB:SOURce:DPLus?
```

The :TRIGger:USB:SOURce:DPLus? query returns the current source for the USB D+ signal.

**Return Format**

```
<source><NL>
```

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:USB:SOURce:DMINus" on page 432
- ":TRIGger:USB:TRIGger" on page 435
3  Commands by Subsystem

:TRIGger:USB:SPEed

(see page 564)

Command Syntax

:TRIGger:USB:SPEed <value>

<value> ::= {LOW | FULL}

The :TRIGger:USB:SPEed command sets the expected USB signal speed to be Low Speed (1.5 Mb/s) or Full Speed (12 Mb/s).

Query Syntax

:TRIGger:USB:SPEed?

The :TRIGger:USB:SPEed? query returns the current speed value for the USB signal.

Return Format

<value><NL>

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:USB:SOURce:DMINus" on page 432
- ":TRIGger:USB:SOURce:DPLus" on page 433
- ":TRIGger:USB:TRIGger" on page 435
:TRIGger:USB:TRIGger

Command Syntax
:TRIGger:USB:TRIGger <value>

<value> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet}

The :TRIGger:USB:TRIGger command sets where the USB trigger will occur:
- SOP — Start of packet.
- EOP — End of packet.
- ENTersuspend — Enter suspend state.
- EXITsuspend — Exit suspend state.
- RESet — Reset complete.

Query Syntax
:TRIGger:USB:TRIGger?

The :TRIGger:USB:TRIGger? query returns the current USB trigger value.

Return Format
<value><NL>

<value> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet}

See Also
- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:USB:SPEed" on page 434
# :WAVeform Commands

Provide access to waveform data. See "Introduction to :WAVeform Commands" on page 438.

## Table 76 :WAVeform Commands Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>:WAVeform:BYTeorder</td>
<td>:WAVeform:BYTeorder? (see page 444)</td>
<td>&lt;value&gt; ::= (LSBFirst</td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:COUNt? (see page 445)</td>
<td>&lt;count&gt; ::= an integer from 1 to 65536 in NR1 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:DATA? (see page 446)</td>
<td>&lt;binary block length bytes&gt;, &lt;binary data&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, to transmit 1000 bytes of data, the syntax would be:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#80001000&lt;1000 bytes of data&gt;&lt;NL&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 is the number of digits that follow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00001000 is the number of bytes to be transmitted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;1000 bytes of data&gt; is the actual data</td>
</tr>
<tr>
<td>:WAVeform:FORMat</td>
<td>:WAVeform:FORMat? (see page 448)</td>
<td>&lt;value&gt; ::= (WORD</td>
</tr>
<tr>
<td>:WAVeform:POINts</td>
<td>:WAVeform:POINts? (see page 449)</td>
<td>&lt;# points&gt; ::= (100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;# points&gt; ::= (100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;points_mode&gt; ::= (NORMal</td>
</tr>
<tr>
<td>:WAVeform:POINts:MODE</td>
<td>:WAVeform:POINts:MODE? (see page 452)</td>
<td>&lt;points_mode&gt; ::= (NORMal</td>
</tr>
</tbody>
</table>
## Table 76 :WAVeform Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:WAVeform:PREamble? (see page 453)</td>
<td><code>&lt;preamble_block&gt; ::= &lt;format NR1&gt;,&lt;type NR1&gt;,&lt;points NR1&gt;,&lt;count NR1&gt;,&lt;xincrement NR3&gt;,&lt;xorigin NR3&gt;,&lt;xreference NR1&gt;,&lt;yincrement NR3&gt;,&lt;yorigin NR3&gt;,&lt;yreference NR1&gt;</code>&lt;br&gt;<code>&lt;format&gt; ::= an integer in NR1 format:</code>&lt;br&gt;<code>• 0 for BYTE format</code>&lt;br&gt;<code>• 1 for WORD format</code>&lt;br&gt;<code>• 2 for ASCII format</code>&lt;br&gt;<code>&lt;type&gt; ::= an integer in NR1 format:</code>&lt;br&gt;<code>• 0 for NORMAL type</code>&lt;br&gt;<code>• 1 for PEAK detect type</code>&lt;br&gt;<code>• 2 for AVERAGE type</code>&lt;br&gt;<code>• 3 for HRESolution type</code>&lt;br&gt;<code>&lt;count&gt; ::= Average count, or 1 if PEAK detect type or NORMAL; an integer in NR1 format</code></td>
</tr>
<tr>
<td></td>
<td>:WAVeform:SOURce &lt;source&gt; (see page 456)</td>
<td>`&lt;source&gt; ::= {CHANnel&lt;n&gt;</td>
</tr>
<tr>
<td></td>
<td>:WAVeform:TYPE? (see page 460)</td>
<td>`&lt;return_mode&gt; ::= {NORM</td>
</tr>
<tr>
<td></td>
<td>:WAVeform:UNSIGNED {{0</td>
<td>OFF}</td>
</tr>
<tr>
<td></td>
<td>:WAVeform:VIEW &lt;view&gt; (see page 462)</td>
<td><code>&lt;view&gt; ::= {MAIN}</code></td>
</tr>
<tr>
<td></td>
<td>:WAVeform:XINCrement? (see page 463)</td>
<td><code>&lt;return_value&gt; ::= x-increment in the current preamble in NR3 format</code></td>
</tr>
<tr>
<td></td>
<td>:WAVeform:XORigin? (see page 464)</td>
<td><code>&lt;return_value&gt; ::= x-origin value in the current preamble in NR3 format</code></td>
</tr>
<tr>
<td></td>
<td>:WAVeform:XREFERENCE? (see page 465)</td>
<td><code>&lt;return_value&gt; ::= 0 (x-reference value in the current preamble in NR1 format)</code></td>
</tr>
<tr>
<td></td>
<td>:WAVeform:YINCrement? (see page 466)</td>
<td><code>&lt;return_value&gt; ::= y-increment value in the current preamble in NR3 format</code></td>
</tr>
</tbody>
</table>
Table 76  :WAVeform Commands Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Query</th>
<th>Options and Query Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>:WAVeform:YORigin? (see page 467)</td>
<td>&lt;return_value&gt; ::= y-origin in the current preamble in NR3 format</td>
</tr>
<tr>
<td>n/a</td>
<td>:WAVeform:YREFerence? (see page 468)</td>
<td>&lt;return_value&gt; ::= y-reference value in the current preamble in NR1 format</td>
</tr>
</tbody>
</table>

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 446) and :WAVeform:PREamble (see page 453). 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 three types of waveform acquisitions that can be selected for analog channels with the :ACQuire:TYPE command (see page 148): NORMal, AVERage, PEAK, and HRESolution. Digital channels are always acquired using NORMal. When the data is acquired using the :DIGitize command (see page 111) or :RUN command (see page 131), 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 GPIB 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 446) before changing any instrument setup.

A waveform record consists of either all of the acquired points or a subset of the acquired points. The number of points acquired may be queried using :ACQuire:POINts? (see page 145).

Helpful Hints:
The number of points transferred to the computer is controlled using the :WAVEform:POINts command (see page 449). 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 GPIB 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 449). 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 142). 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 449). 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 449). 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 148), the value returned by the :WAVEform:XINCrement query (see page 463) should be doubled to find the time difference between the min-max pairs.

**HRESolution Data**

The high resolution (*smoothing*) mode is used to reduce noise at slower sweep speeds where the digitizer samples faster than needed to fill memory for the displayed time range. This mode is the same as the AVERage mode with :ACQuire:COUNt 1.

**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:

\[
\text{voltage} = [(\text{data value} - \text{yreference}) \times \text{yincrement}] + \text{yorigin}
\]

If the :WAVEform:FORMat data format is ASCii (see page 448), 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 \texttt{:WAVE:FORMat} = 16 ns, \texttt{:WAVE:FORM:INC} = 0, and \texttt{:WAVE:FORM:ORIGIN} = 2 ns, can be calculated using the following formula:

\[
\text{time} = [(\text{data point number} - \text{xreference}) \times \text{xincrement}] + \text{xorigin}
\]

This would result in the following calculation for time bucket 3:

\[
\text{time} = [(3 - 0) \times 2 \text{ ns}] + 16 \text{ ns} = 22 \text{ ns}
\]

In :ACQ:TYPE PEAK mode (see page 148), because data is acquired in max-min pairs, modify the previous time formula to the following:

\[
\text{time} = [(\text{data pair number} - \text{xreference}) \times \text{xincrement} \times 2] + \text{xorigin}
\]

**Data Format for Transfer**

There are three formats for transferring waveform data over the interface: BYTE, WORD and ASCii (see "\texttt{:WAVE:FORMat}" on page 448). 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 ",\texttt{#8000001000}" would be sent. The 8 indicates that eight length bytes follow, and 00001000 indicates that 1000 binary data bytes follow.

Use the \texttt{:WAVE:UNSigned} command (see page 461) 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 "\texttt{:WAVE:FORMat}" on page 448) 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 \texttt{:WAVE:BYT} (see page 444) and \texttt{:WAVE:UNS} (see page 461) have no effect when the format is ASCii.
Data Format for Transfer - WORD format

WORD format (see ":WAVeform:FORMat" on page 448) 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 449). 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 444) 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 448) 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 GPIB 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 444) has no effect when the data format is BYTE.

Digital Channel Data (MSO models only)

The waveform record for digital channels is similar to that of analog channels. The main difference is that the data points represent either DIGital0,..,7 (POD1), DIGital8,..,15 (POD2), or any grouping of digital channels (BUS1 or BUS2).

For digital channels, :WAVeform:UNSsigned (see page 461) 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:

<table>
<thead>
<tr>
<th>:WAVEform:SOURce</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>POD1</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>POD2</td>
<td>D15</td>
<td>D14</td>
<td>D13</td>
<td>D12</td>
<td>D11</td>
<td>D10</td>
<td>D9</td>
<td>D8</td>
</tr>
</tbody>
</table>

If the :WAVEform:FORMat is WORD (see page 448) is WORD, every other data byte will be 0. The setting of :WAVEform:BYTecorder (see page 444) 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 150) 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 WORD;POIN +1000;SOUR CHAN1
```
### :WAVeform:BYTeorder

(see page 564)

**Command Syntax**

:WAVeform:BYTeorder <value>

<value> ::= {LSBFirst | MSBFirst}

The :WAVeform:BYTeorder command sets the output sequence of the WORD data. The parameter MSBFirst sets the most significant byte to be transmitted first. The parameter LSBFirst sets the least significant byte to be transmitted first. This command affects the transmitting sequence only when :WAVeform:FORMat WORD is selected. The default setting is LSBFirst.

**Query Syntax**

:WAVeform:BYTeorder?

The :WAVeform:BYTeorder query returns the current output sequence.

**Return Format**

<value><NL>

<value> ::= {LSBF | MSBF}

**See Also**

- "Introduction to :WAVeform Commands" on page 438
- ":WAVeform:DATA" on page 446
- ":WAVeform:FORMat" on page 448
- ":WAVeform:PReamble" on page 453

**Example Code**

- "Example Code" on page 457
- "Example Code" on page 454
:WAVeform:COUNt

(see page 564)

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 438
- ":ACQuire:COUNt" on page 142
- ":ACQuire:TYPE" on page 148
**:WAVeform:DATA**

(see page 564)

**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. Holes can be reasonably expected in the equivalent time acquisition mode (especially at slower horizontal sweep speeds when measuring low frequency signals).

Another situation where there can be zeros in the data, incorrectly, is when programming over telnet port 5024. Port 5024 provides a command prompt and is intended for ASCII transfers. Use telnet port 5025 instead.

- **0x01 or 0x0001** — Clipped low. These are locations where the waveform is clipped at the bottom of the oscilloscope display.

- **0xFF or 0xFFFF** — Clipped high. These are locations where the waveform is clipped at the top of the oscilloscope display.

**Return Format**

```
<binary block data><NL>
```

**See Also**

- "Introduction to :WAVeform Commands" on page 438
- ":WAVeform:UNSigned" on page 461
- ":WAVeform:BYTeorder" on page 444
- ":WAVeform:FORMat" on page 448
- ":WAVeform:POINts" on page 449
- ":WAVeform:PREamble" on page 453
- ":WAVeform:SOURce" on page 456
- ":WAVeform:TYPE" on page 460

**Example Code**

```
' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString "::WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
```
' 
' <header><waveform_data><NL>
'
' Where:
' <header> = #800001000 (This is an example header)
' The "#8" may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block. The
' size can vary depending on the number of points acquired for the
' waveform. You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
' Unsigned integer bytes.
For lngI = 0 To UBound(varQueryResult) _
    Step (UBound(varQueryResult) / 20) ' 20 points.
    If intBytesPerData = 2 Then
        lngDataValue = varQueryResult(lngI) * 256 _
        + varQueryResult(lngI + 1) ' 16-bit value.
    Else
        lngDataValue = varQueryResult(lngI) ' 8-bit value.
    End If
    strOutput = strOutput + "Data point " + _
    CStr(lngI / intBytesPerData) + ", " + _
    FormatNumber((lngDataValue - lngYReference) _
    * sngYIncrement + sngYOrigin) + " V," + _
    FormatNumber(((lngI / intBytesPerData - lngXReference) _
    * sngXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

Example program from the start: "VISA COM Example in Visual Basic" on
page 614
### :WAVeform:FORMat

(see page 564)

**Command Syntax**

```plaintext
: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.

  When the :WAVeform:SOURce is the serial decode bus (SBUS), ASCII is the only waveform format allowed.

- BYTE formatted data is transferred as 8-bit bytes.

  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**

```plaintext
:WAVeform:FORMat?
```

The :WAVeform:FORMat query returns the current output format for the transfer of waveform data.

**Return Format**

```plaintext
<value><NL>
```

<value> ::= {WORD | BYTE | ASC}

**See Also**

- "Introduction to :WAVeform Commands" on page 438
- ":WAVeform:BYTEorder" on page 444
- ":WAVeform:DATA" on page 446
- ":WAVeform:PREamble" on page 453

**Example Code**

- "Example Code" on page 457
:WAVeform:POINts

(see page 564)

Command Syntax

:WAVeform:POINts <# points>

<# points> ::= (100 | 250 | 500 | 1000 | <points mode>)
if waveform points mode is NORMAL

<# points> ::= (100 | 250 | 500 | 1000 | 2000 ... 8000000
in 1-2-5 sequence | <points mode>)
if waveform points mode is MAXimum or RAW

<points mode> ::= (NORMAL | MAXimum | RAW)

NOTE

The <points_mode> option is deprecated. Use the :WAVeform:POINts:MODE command instead.

The :WAVeform:POINts command sets the number of waveform points to be transferred with the :WAVeform:DATA? query. This value represents the points contained in the waveform selected with the :WAVeform:SOURce command.

For the analog or digital sources, 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 and may be up to 8,000,000. The raw acquisition record can only be transferred when the oscilloscope is not running and can only be retrieved from the analog or digital sources.

- The second is referred to as the measurement record and is a 1000 point (maximum) representation of the raw acquisition record. The measurement record can be retrieved at any time, from any source.

See the :WAVeform:POINts:MODE command (see page 451) for more information on the <points_mode> option.

Only data visible on the display will be returned.

The maximum number of points returned when the waveform source is math or function is 1000.

When the :WAVeform:SOURce is the serial decode bus (SBUS), 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 451) for more information).
When the :WAVeform:SOURce is the serial decode bus (SBUS), this query returns the number of messages that were decoded.

**Return Format**

```
<# points><NL>
<# points> ::= {100 | 250 | 500 | 1000 | <maximum # points>}
if waveform points mode is NORMal
<# points> ::= {100 | 250 | 500 | 1000 | 2000 ... 8000000
in 1-2-5 sequence | <maximum # points>}
if waveform points mode is MAXimum or RAW
```

**NOTE**

If a full screen of data is not displayed, the number of points returned will not be 1000 or an even divisor of it.

**See Also**

- "Introduction to :WAVeform Commands" on page 438
- ":ACQuire:POINts" on page 145
- ":WAVeform:DATA" on page 446
- ":WAVeform:SOURce" on page 456
- ":WAVeform:VIEW" on page 462
- ":WAVeform:PREamble" on page 453
- ":WAVeform:POINts:MODE" on page 451

**Example Code**

```
' WAVE_POINTS - Specifies the number of points to be transferred
' using the ":WAVEFORM:DATA?" query.
myScope.WriteString ":WAVEFORM:POINTS 1000"
```

Example program from the start: "VISA COM Example in Visual Basic" on page 614
:**WAVEform:**POINts:**MODE**

(see page 564)

**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 transferred when the oscilloscope is not running and can only be retrieved from the analog or digital sources.

- The second is referred to as the measurement record and is a 1000 point (maximum) representation of the raw acquisition record. The measurement record can be retrieved at any time, from any source.

If the <points_mode> is NORMal, the measurement record is retrieved.

If the <points_mode> is RAW, the raw acquisition record is used. Under some conditions, such as when the oscilloscope is running, 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, if the raw acquisition record is unavailable (for example, when the oscilloscope is running), or if the reconstruction filter (\(\text{Sin}(x)/x\) interpolation) is in use, the measurement record may have more data. If data is being retrieved as the oscilloscope is stopped and as the data being retrieved can switch between the measurement and raw acquisition records.

Considerations for MAXimum or RAW data retrieval

- The instrument must be stopped (see the :STOP command (see page 135) or the :DIGitize command (see page 111) in the root subsystem) in order to return more than 1000 points.

- :TIMebase:MODE must be set to MAIN.

- :ACQuire:TYPE must be set to NORMal, AVERage, or HRESolution. If AVERage, :ACQuire:COUNt must be set to 1 in order to return more than 1000 points.

- MAXimum or RAW will allow up to 8,000,000 points to be returned. The number of points returned will vary as the instrument's configuration is changed. Use the :WAVEform:POINts? MAXimum query to determine the maximum number of points that can be retrieved at the current settings.
Query Syntax :WAVeform:POINts:MODE?

The :WAVeform:POINts:MODE? query returns the current points mode. Setting the points mode will affect what data is transferred. See the discussion above.

Return Format <points_mode><NL>

See Also
- "Introduction to :WAVeform Commands" on page 438
- ":ACQuire:POINts" on page 145
- ":WAVEform:DATA" on page 446
- ":WAVEform:VIEW" on page 462
- ":WAVEform:PREamble" on page 453
- ":WAVEform:POINts" on page 449
- ":TIMebase:MODE" on page 322
- ":ACQuire:TYPE" on page 148
- ":ACQuire:COUNt" on page 142
:WAVEform:PREamble

(see page 564)

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, 2 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 438
- ":ACQuire:COUNt" on page 142
- ":ACQuire:POINts" on page 145
- ":ACQuire:TYPE" on page 148
- ":DIGitize" on page 111
- ":WAVEform:COUNt" on page 445
- ":WAVEform:DATA" on page 446
- ":WAVEform:FORMAT" on page 448
- ":WAVEform:POINts" on page 449
- ":WAVEform:TYPE" on page 460
- ":WAVEform:XINCrement" on page 463
- ":WAVEform:XORigin" on page 464
- ":WAVEform:XREFerence" on page 465
- ":WAVEform:YINCrement" on page 466
- ":WAVEform:YORigin" on page 467
- ":WAVEform:YREFerence" on page 468

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, 2 = ASCII.
```
'TYPE': int16 - 0 = NORMAL, 1 = PEAK DETECT, 2 = AVERAGE

'POINTS': int32 - number of data points transferred.

'COUNT': int32 - 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

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)

Example program from the start: "VISA COM Example in Visual Basic" on page 614
3 Commands by Subsystem

:WAVEform:SOURce

(see page 564)

Command Syntax

:WAVEform:SOURce <source>

<source> ::= {CHANnel<n> | FUNCTION | MATH | SBUS} for DSO models

<source> ::= {CHANnel<n> | POD{1 | 2} | BUS{1 | 2} | FUNCTION
 | MATH | SBUS} for MSO models

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

<n> ::= {1 | 2} for the two channel oscilloscope models

The :WAVEform:SOURce command selects the analog channel, function, digital pod, digital bus, 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.

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 448).

  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 timestamps and hexadecimal bus values, for example:
  
  -5.000000000000e-08,0x1938,-4.990000000000e-08,0xff38,...

Query Syntax

:WAVEform:SOURce?

The :WAVEform:SOURce? query returns the currently selected source for the WAVEform commands.

NOTE

MATH is an alias for FUNCTION. The :WAVEform:SOURce? Query returns FUNC if the source is FUNCTION or MATH.

Return Format

<source><NL>
<source> ::= {CHAN<n> | FUNC | SBUS} for DSO models
<source> ::= {CHAN<n> | POD(1 | 2) | BUS(1 | 2) | FUNC | SBUS}
for MSO models
<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

See Also
- "Introduction to :WAVeform Commands" on page 438
- ":DIGitize" on page 111
- ":WAVEform:FORMat" on page 448
- ":WAVEform:BYTеorder" on page 444
- ":WAVEform:DATA" on page 446
- ":WAVEform:PREamble" on page 453

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, 2 = 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) + 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 = " +
Commands by Subsystem

FormatNumber(((lngPoints / 2 - lngXReference) * _
   dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString ":\WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
'
' <header><waveform_data><NL>
'
' Where:
' <header> = #800001000 (This is an example header)
' The "#8" may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block. The
' size can vary depending on the number of points acquired for the
' waveform. You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

' Unsigned integer bytes.
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)

For lngI = 0 To UBound(varQueryResult) _
   Step (UBound(varQueryResult) / 20) ' 20 points.
   If intBytesPerData = 2 Then
      lngDataValue = varQueryResult(lngI) * 256 _
      + varQueryResult(lngI + 1) ' 16-bit value.
   Else
      lngDataValue = varQueryResult(lngI) ' 8-bit value.
   End If
   strOutput = strOutput + "Data point " + _
      CStr(lngI / intBytesPerData) + ", " + 
      FormatNumber((lngDataValue - lngYReference) _
      * sngYIncrement + sngYOrigin) + " V," + _
      FormatNumber(((lngI / intBytesPerData - lngXReference) _
      * sngXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

Example program from the start: "VISA COM Example in Visual Basic" on
page 614
**:WAVEform:TYPE**

(see page 564)

**Query Syntax**

:WAVEform:TYPE?

The :WAVEform:TYPE? query returns the acquisition mode associated with the currently selected waveform. The acquisition mode is set by the :ACQuire:TYPE command.

**Return Format**

```
<mode><NL>
```

<mode> ::= {NORM | PEAK | AVER | HRES}

**NOTE**

If the :WAVEform:SOURce is POD1, POD2, or SBUS, the type is always NORM.

**See Also**

- "Introduction to :WAVEform Commands" on page 438
- ":ACQuire:TYPE" on page 148
- ":WAVEform:DATA" on page 446
- ":WAVEform:PREamble" on page 453
- ":WAVEform:SOURce" on page 456
**:WAVeform:UNSIGNED**

(see page 564)

**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 or POD2, 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 438
- ":WAVeform:SOURce" on page 456
:WAVeform:VIEW

(see page 564)

Command Syntax

:WAVeform:VIEW <view>

<view> ::= {MAIN}

The :WAVeform:VIEW command sets the view setting associated with the currently selected waveform. Currently, the only legal value for the view setting is MAIN.

Query Syntax

:WAVeform:VIEW?

The :WAVeform:VIEW? query returns the view setting associated with the currently selected waveform.

Return Format

:view><NL>

<view> ::= {MAIN}

See Also

• "Introduction to :WAVeform Commands" on page 438
• ":WAVeform:POINts" on page 449
:WAVeform:XINCrement

(see page 564)

**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 438
- "::WAVeform:PREamble" on page 453

**Example Code**

- "Example Code" on page 454
:WAVEform:XORigin

(see page 564)

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 438
- ":WAVEform:PREamble" on page 453
- ":WAVEform:XREFerence" on page 465

Example Code

- "Example Code" on page 454
**:WAVeform:XREFerence**

(see page 564)

**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 438
- ":WAVeform:PREamble" on page 453
- ":WAVeform:XORigin" on page 464

**Example Code**

- "Example Code" on page 454
**:WAVeform:YINCrement**

(see page 564)

**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 438
- ":WAVeform:PREamble" on page 453

**Example Code**
- "Example Code" on page 454
:WAVEform:YORigin

(see page 564)

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 438
- ":WAVEform:PREamble" on page 453
- ":WAVEform:YREFerence" on page 468

Example Code
- "Example Code" on page 454
**:WAVeform:YREFerence**

(see page 564)

**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 438
- ":WAVeform:PREamble" on page 453
- ":WAVeform:YORigin" on page 467

**Example Code**

- "Example Code" on page 454
4
Commands A-Z

A  469
B  470
C  471
D  473
E  474
F  475
G  476
H  476
I  476
L  477
M  477
N  479
O  479
P  480
Q  481
R  481
S  482
T  484
U  488
V  489
W  489
X  490
Y  490

A
- AALias, ":ACQuire:AALias" on page 140
- ACKnowledge, ":TRIGger:CAN:ACKnowledge" on page 528
- ":ACQuire:AALias" on page 140
- ":ACQuire:COMComplete" on page 141
- ":ACQuire:COUNt" on page 142
- ":ACQuire:DAALias" on page 143
470 Agilent 6000 Series Oscilloscopes Programmer’s Reference

4 Commands A-Z

• ":ACQuire:MODE" on page 144
• ":ACQuire:POINts" on page 145
• ":ACQuire:RSIGnal" on page 146
• ":ACQuire:SRATe" on page 147
• ":ACQuire:TYPE" on page 148
• ":ACTivity" on page 103

ADDRess Commands:
• ",SBUS:BUSDoctor:ADDRess" on page 294
• ",TRIGger:IIC:PADTern:ADDRess" on page 392
• ",AER (Arm Event Register)" on page 104

ASIZe, ",SBUS:IIC:ASIZe" on page 308

• ",AUToscale" on page 105
• ",AUToscale:AMODE" on page 107
• ",AUToscale:CHANnels" on page 108

BAUDrate Commands:
• ",SBUS:BUSDoctor:BAUDrate" on page 295
• ",TRIGger:CAN:SIGNal:BAUDrate" on page 350
• ",TRIGger:LIN:SIGNal:BAUDrate" on page 403
• BIT<m>, ",BUS<n>:BIT<m>" on page 152
• BITS, ",BUS<n>:BITS" on page 153
• ",BLANk" on page 109
• ",BUS<n>:BIT<m>" on page 152
• ",BUS<n>:BITS" on page 153
• ",BUS<n>:CLEar" on page 155
• ",BUS<n>:DISPLAY" on page 156
• ",BUS<n>:LABel" on page 157
• ",BUS<n>:MASK" on page 158

BUSDoctor Commands:
• ",SBUS:BUSDoctor:ADDRess" on page 294
• ",SBUS:BUSDoctor:BAUDrate" on page 295
• ",SBUS:BUSDoctor:CHANnel" on page 296
• ",SBUS:BUSDoctor:MODE" on page 297

BWLimit Commands:
• ",CHANnel<n>:BWLimit" on page 170
• ",EXTernal:BWLimit" on page 204
BYTeorder, "WAVEform:BYTeorder" on page 444

CALibrate:DATE" on page 160
CALibrate:LABel" on page 161
CALibrate:START" on page 162
CALibrate:STATus" on page 163
CALibrate:SWITCh" on page 164
CALibrate:TEMPerature" on page 165
CALibrate:TIME" on page 166

CAN Commands:
- "SBUS:CAN:COUNt:ERRor" on page 298
- "SBUS:CAN:COUNt:OVERload" on page 299
- "SBUS:CAN:COUNt:RESet" on page 300
- "SBUS:CAN:COUNt:TOTal" on page 301
- "SBUS:CAN:COUNt:UTILization" on page 302
- "TRIGger:CAN Commands" on page 343
- CBASe, "TRIGger:FLEXray:TIME:CBASe" on page 377
- CCBASe, "TRIGger:FLEXray:FRAME:CCBase" on page 373
- CCREpetition, "TRIGger:FLEXray:FRAME:CCRepetition" on page 374
- CRepetition, "TRIGger:FLEXray:TIME:CREPetition" on page 378
- "CDISplay" on page 110
- CENTer, "FUNCTION:CENTer" on page 214
- CHANnel, "SBUS:BUSDoctor:CHANnel" on page 296
- "CHANnel:ACTivity" on page 496
- "CHANnel:LABel" on page 497
- "CHANnel:THReshold" on page 498
- "CHANnel2:SKEW" on page 499
- "CHANnel<n>:BWLimit" on page 170
- "CHANnel<n>:COUPLing" on page 171
- "CHANnel<n>:DISPlay" on page 172
- "CHANnel<n>:IMPedance" on page 173
- "CHANnel<n>:INPut" on page 500
- "CHANnel<n>:INVert" on page 174
- "CHANnel<n>:LABel" on page 175
- "CHANnel<n>:OFFSet" on page 176
- "CHANnel<n>:PMODe" on page 501
- ".:CHANnel<n>:PROBe" on page 177
- ".:CHANnel<n>:PROBe:ID" on page 178
- ".:CHANnel<n>:PROBe:SKEW" on page 179
- ".:CHANnel<n>:PROBe:STYPe" on page 180
- ".:CHANnel<n>:PROTection" on page 181
- ".:CHANnel<n>:RANGe" on page 182
- ".:CHANnel<n>:SCALE" on page 183
- ".:CHANnel<n>:UNITs" on page 184
- ".:CHANnel<n>:VERNier" on page 185

CLEar Commands:
- ".:BUS<n>:CLEar" on page 155
- ".:DISPlay:CLEar" on page 195
- ".:MEASure:CLEar" on page 250

CLOck Commands:
- ".:TRIGger:IIC:SOURce:CLOCk" on page 395
- ".:TRIGger:SPI:CLOCk:SLOPe" on page 417
- ".:TRIGger:SPI:CLOCk:TIMEout" on page 418
- ".:TRIGger:SPI:SOURce:CLOCk" on page 422
- ".:*CLS (Clear Status)" on page 79

COMplete, ".:ACQuire:COMplete" on page 141

CONNect, ".:DISPlay:CONNect" on page 502

COUNt Commands:
- ".:ACQuire:COUNt" on page 142
- ".:SBUS:CAN:COUNt:ERRor" on page 298
- ".:SBUS:CAN:COUNt:OVERload" on page 299
- ".:SBUS:CAN:COUNt:RESet" on page 300
- ".:SBUS:CAN:COUNt:TOTal" on page 301
- ".:SBUS:CAN:COUNt:UTILization" on page 302
- ".:SBUS:FLEXray:COUNt:NULL" on page 304
- ".:SBUS:FLEXray:COUNt:RESet" on page 305
- ".:SBUS:FLEXray:COUNt:SYNC" on page 306
- ".:SBUS:FLEXray:COUNt:TOTal" on page 307
- ".:TRIGger:EBURst:COUNt" on page 361
- ".:WAVeform:COUNt" on page 445

COUNter, ".:MEASure:COUNter" on page 251
COUPling Commands:
- ":CHANnel<n>:COUPling" on page 171
- ":TRIGger[:EDGE]:COUPling" on page 365

DAALias, ":ACQuire:DAALias" on page 143

DATA Commands:
- ":DISPlay:DATA" on page 196
- ":TRIGger:CAN:PA TTern:DATA*" on page 345
- ":TRIGger:CAN:PA TTern:DATA:LENGth" on page 346
- ":TRIGger:IIC:PA TTern:DATA" on page 393
- ":TRIGger:IIC:PA TTern:DATA2" on page 394
- ":TRIGger:IIC:SOURce:DATA*" on page 396
- ":TRIGger:SPI:PA TTern:DATA" on page 420
- ":TRIGger:SPI:SOURce:DATA" on page 423
- ":WAVeform:DATA" on page 446

DATE Commands:
- ":CA Librate:DATE" on page 160
- ":SYSTem:DATE" on page 313

DEFine, ":MEASure:DEFine" on page 252

DEFinition Commands:
- ":TRIGger:CAN:SIGNal:DEFinition" on page 529
- ":TRIGger:LIN:SIGNal:DEFinition" on page 530

DELay Commands:
- ":MEASure:DE Lay" on page 255
- ":TI Mebase:DE Lay" on page 527

DESTination, ":HARDcopy:DESTination" on page 508

DEVice, ":HARDcopy:DEVice" on page 509

"DIGital<n>:DISPlay" on page 188

"DIGital<n>:LABEL" on page 189

"DIGital<n>:PO Sition" on page 190

"DIGital<n>:SIZE" on page 191

"DIGital<n>:TH REshold" on page 192

"DIGitize" on page 111

DISPlay Commands:
- ":BUS<n>:DISPlay" on page 156
- ":CHANnel<n>:DISPlay" on page 172
Commands A-Z

- ":DIGital<n>:DISPlay" on page 188
- ":FUNCTION:DISPlay" on page 215
- ":POD<n>:DISPlay" on page 288
- ":SBUS:DISPlay" on page 303
- ":DISPlay:CLEar" on page 195
- ":DISPlay:CONNect" on page 502
- ":DISPlay:DATA" on page 196
- ":DISPlay:LABel" on page 198
- ":DISPlay:LABList" on page 199
- ":DISPlay:ORDer" on page 503
- ":DISPlay:PERSistence" on page 200
- ":DISPlay:SOURce" on page 201
- ":DISPlay:VECTors" on page 202
- DMINus, ":TRIGger:USB:SOURce:DMINus" on page 432
- DPLus, ":TRIGger:USB:SOURce:DPLus" on page 433
- DSP, ":SYSTem:DSP" on page 314
- DURation, ":TRIGger:DURation Commands" on page 354
- DUTYcycle, ":MEASure:DUTYcycle" on page 257
- EBURst, ":TRIGger:EBURst Commands" on page 360
- EDGE, ":TRIGger[:EDGE] Commands" on page 364
- ":ERASe" on page 504

ERRor Commands:
- ":SBUS:CAN:COUNt:ERRor" on page 298
- ":SYSTem:ERRor" on page 315
- ":TRIGger:FLEXray:ERRor:TYPE" on page 371
- "*ESE (Standard Event Status Enable)" on page 80
- "*ESR (Standard Event Status Register)" on page 82
- ":EXTernal:BWLimit" on page 204
- ":EXTernal:IMPedance" on page 205
- ":EXTernal:INPut" on page 505
- ":EXTernal:PMODE" on page 506
- ":EXTernal:PROBe" on page 206
- ":EXTernal:PROBe:ID" on page 207
- ":EXTernal:PROBe:STYPe" on page 208
- ":EXTernal:PROTection" on page 209
- ":EXTernal:RANGe" on page 210
- ":EXTernal:UNITs" on page 211

F
- FACTors, ":HARDcopy:FACTors" on page 225
- FALLtime, ":MEASure:FALLtime" on page 258
- FFEed, ":HARDcopy:FFEed" on page 226
- FILEname, ":HARDcopy:FILEname" on page 227
- FIND, ":TRIGger:SEQUence:FIND" on page 411
- FLEXray Commands:
  - ":SBUS:FLEXray:COUNt:NULL" on page 304
  - ":SBUS:FLEXray:COUNt:RESet" on page 305
  - ":SBUS:FLEXray:COUNt:SYNC" on page 306
  - ":SBUS:FLEXray:COUNt:TOTal" on page 307
  - ":TRIGger:FLEXray:ERRor:TYPE" on page 371
  - ":TRIGger:FLEXray:FRAME:CCBase" on page 373
  - ":TRIGger:FLEXray:FRAME:CCRepetition" on page 374
  - ":TRIGger:FLEXray:FRAME:ID" on page 375
  - ":TRIGger:FLEXray:FRAME:TYPE" on page 376
  - ":TRIGger:FLEXray:TIME:CBASE" on page 377
  - ":TRIGger:FLEXray:TIME:CREPetition" on page 378
  - ":TRIGger:FLEXray:TIME:SEGMENT" on page 379
  - ":TRIGger:FLEXray:TIME:SLOT" on page 380
  - ":TRIGger:FLEXray:TRIGger" on page 381
- FORMat Commands:
  - ":HARDcopy:FORMat" on page 228
  - ":WAVeform:FORMat" on page 448
- FRAMe Commands:
  - ":TRIGger:FLEXray:FRAME:CCBase" on page 373
  - ":TRIGger:FLEXray:FRAME:CCRepetition" on page 374
  - ":TRIGger:FLEXray:FRAME:ID" on page 375
  - ":TRIGger:FLEXray:FRAME:TYPE" on page 376
  - ":TRIGger:SPI:SOURce:FRAME" on page 424
- FRAMing, ":TRIGger:SPI:FRAMing" on page 419
- FREQuency, ":MEASure:FREQuency" on page 259
- ":FUNCtion:CENTer" on page 214
- ":FUNCtion:DISPlay" on page 215
Commands A-Z

- ":FUNCTION:OFFSet" on page 216
- ":FUNCTION:OPERation" on page 217
- ":FUNCTION:RANGe" on page 218
- ":FUNCTION:REFERence" on page 219
- ":FUNCTION:SCALe" on page 220
- ":FUNCTION:SOURce" on page 221
- ":FUNCTION:SPAN" on page 222
- ":FUNCTION:VIEW" on page 507
- ":FUNCTION:WINDow" on page 223

G
- GLITch (Pulse Width), ":TRIGger:GLITch Commands" on page 382
- GRAYscale, ":HARDcopy:GRAYscale" on page 510
- GREaterthan Commands:
  - ":TRIGger:DURation:GREaterthan" on page 355
  - ":TRIGger:GLITch:GREaterthan" on page 384

H
- ":HARDcopy:DESTination" on page 508
- ":HARDcopy:DEVice" on page 509
- ":HARDcopy:FACTors" on page 225
- ":HARDcopy:FFEed" on page 226
- ":HARDcopy:FILENAME" on page 227
- ":HARDcopy:FORMAT" on page 228
- ":HARDcopy:GRAYscale" on page 510
- ":HARDcopy:IGColors" on page 229
- ":HARDcopy:PALette" on page 230
- ":HARDcopy:PDRiver" on page 231
- HFReject, ":TRIGger:HFReject" on page 336
- HOLDoff, ":TRIGger:HOLDoff" on page 337

I
- ID Commands:
  - ":TRIGger:CAN:PATTer:nID" on page 347
  - ":TRIGger:CAN:PATTer:nID:MODE" on page 348
  - ":TRIGger:FLEXray:FRAMe:ID" on page 375
- IDLE, ":TRIGger:EBURst:IDLE" on page 362
- "*IDN (Identification Number)" on page 84
- IIC Commands:
  - ":SBUS:IIC:ASIZE" on page 308
- ".TRIGger:IIC Commands" on page 391
- IGColors, ".HARDcopy:IGColors" on page 229
- IMPedance Commands:
  - ".CHANnel<n>:IMPedance" on page 173
  - ".EXTERNal:IMPedance" on page 205
- INVert, ".CHANnel<n>:INVert" on page 174

L
- LABel Commands:
  - ".BUS<n>:LABel" on page 157
  - ".CALibrate:LABel" on page 161
  - ".CHANnel:LABel" on page 497
  - ".CHANnel<n>:LABel" on page 175
  - ".DIGital<n>:LABel" on page 189
  - ".DISPlay:LABel" on page 198
- LABList, ".DISPlay:LABList" on page 199
- LENGth, ".TRIGger:CAN:PA TTern:DATA:LENGth" on page 346
- LESSthan Commands:
  - ".TRIGger:DURation:LESSthan" on page 356
  - ".TRIGger:GLITch:LESSthan" on page 385
- LEVel Commands:
  - ".TRIGger[:EDGE]:LEVel" on page 366
  - ".TRIGger:GLITch:LEVel" on page 386
- LIN Commands:
  - ".SBUS:LIN:PARity" on page 309
  - ".TRIGger:LIN Commands" on page 400
- LINE, ".TRIGger:TV:LINE" on page 426
- LOCK, ".SYSTem:LOCK" on page 316
- LOWer, ".MEASure:LOWer" on page 511
- "*LRN (Learn Device Setup)" on page 85

M
- ".MARKer:MODE" on page 234
- ".MARKer:X1Position" on page 235
- ".MARKer:X1Y1source" on page 236
- ".MARKer:X2Position" on page 237
- ".MARKer:X2Y2source" on page 238
- ".MARKer:XDELta" on page 239
".MARKer:Y1Position" on page 240
".MARKer:Y2Position" on page 241
".MARKer:YDELta" on page 242
MASK, ":BUS<n>:MASK" on page 158
".MEASure:CLEar" on page 250
".MEASure:COUNter" on page 251
".MEASure:DEFine" on page 252
".MEASure:DELay" on page 255
".MEASure:DUTYcycle" on page 257
".MEASure:FAlltime" on page 258
".MEASure:FREQuency" on page 259
".MEASure:LOWer" on page 511
".MEASure:NWIDth" on page 260
".MEASure:OVERshoot" on page 261
".MEASure:PERiod" on page 263
".MEASure:PHASE" on page 264
".MEASure:PREShoot" on page 265
".MEASure:PWIDth" on page 266
".MEASure:RISetime" on page 267
".MEASure:SCRatch" on page 512
".MEASure:SDEViation" on page 268
".MEASure:SHOW" on page 269
".MEASure:SOURce" on page 270
".MEASure:TDELta" on page 513
".MEASure:TEDGE" on page 272
".MEASure:THResholds" on page 514
".MEASure:TMAX" on page 515
".MEASure:TMIN" on page 516
".MEASure:TSTArt" on page 517
".MEASure:TSTOP" on page 518
".MEASure:TVALue" on page 274
".MEASure:TVOLT" on page 519
".MEASure:UPPer" on page 521
".MEASure:VAMPlitude" on page 276
".MEASure:VAVerage" on page 277
• ":MEASure:VBASe" on page 278
• ":MEASure:VDELta" on page 522
• ":MEASure:VMAX" on page 279
• ":MEASure:VMIN" on page 280
• ":MEASure:VPP" on page 281
• ":MEASure:VRMS" on page 282
• ":MEASure:VSTArt" on page 523
• ":MEASure:VSTOp" on page 524
• ":MEASure:VTIMe" on page 283
• ":MEASure:VTOP" on page 284
• ":MEASure:XMAX" on page 285
• ":MEASure:XMIN" on page 286
• ":MERGe" on page 119

**MODE Commands:**
- ":ACQuire:MODE" on page 144
- ":MARKer:MODE" on page 234
- ":SBUS:BUSDoctor:MODE" on page 297
- ":SBUS:MODE" on page 310
- ":TIMebase:MODE" on page 322
- ":TRIGger:CAN:PATTern:ID:MODE" on page 348
- ":TRIGger:MODE" on page 338
- ":TRIGger:TV:MODE" on page 427
- ":WAVeform:POINts:MODE" on page 451

N
- REJect, ":TRIGger:NREJect" on page 339
- NULL, ":SBUS:FLEXray:COUNt:NULL" on page 304
- NWIDth, ":MEASure:NWIDth" on page 260

O
- OFFSet Commands:
  - ":CHANnel<n>:OFFSet" on page 176
  - ":FUNCtion:OFFSet" on page 216
  - "**OPC (Operation Complete)" on page 86
  - ":OPEE (Operation Status Enable Register)" on page 120
  - OPERation, ":FUNCtion:OPERation" on page 217
  - ":OPERegister:CONDition (Operation Status Condition Register)" on page 122
  - ":OPERegister[:EVENT] (Operation Status Event Register)" on page 124
• "*OPT (Option Identification)" on page 87
• ORDER, ":DISPlay:ORDER" on page 503
• OVERload, ":SBUS:CAN:COUNT:OVERload" on page 299
• OVERshoot, ":MEASure:OVERshoot" on page 261
• "::OVLena (Overload Event Enable Register)" on page 126
• ":OVLReg (Overload Event Register)" on page 128
• PALette, ":HARDcopy:PALETTE" on page 230
• PARity, ":SBUS:LIN:PARity" on page 309
• PATTern Commands:
  • ":TRIGger:CAN:PATTern:DATA" on page 345
  • ":TRIGger:CAN:PATTern:DATA:LENGTH" on page 346
  • ":TRIGger:CAN:PATTern:ID" on page 347
  • ":TRIGger:CAN:PATTern:ID:MODE" on page 348
  • ":TRIGger:DURATION:PATTern" on page 357
  • ":TRIGger:PATTern" on page 340
  • ":TRIGger:SEQUence:PATTern" on page 412
  • ":TRIGger:SPI:PATTern:DATA" on page 420
  • ":TRIGger:SPI:PATTern:WIDTH" on page 421
• PDRiver, ":HARDcopy:PDRiver" on page 231
• PERiod, ":MEASure:PERiod" on page 263
• PERSistence, ":DISPlay:PERSistence" on page 200
• PHASE, ":MEASure:PHASE" on page 264
• PMODE, ":CHAnnel<n>:PMODE" on page 501
• ":POD<n>:DISPlay" on page 288
• ":POD<n>:SIZE" on page 289
• ":POD<n>:THReshold" on page 290
• POINts Commands:
  • ":ACQuire:POINts" on page 145
  • ":WAVEform:POINts" on page 449
  • ":WAVEform:POINts:MODE" on page 451
• POLarity Commands:
  • ":TRIGger:GLITCH:POLarity" on page 387
  • ":TRIGger:TV:POLarity" on page 428
• POSition Commands:
  • ":DIGital<n>:POSition" on page 190
• ":TIMebase:POSition" on page 323
• ":TIMebase:WINDow:POSition" on page 329
• PREamble, ":WAVeform:PREamble" on page 453
• PREShoot, ":MEASure:PREShoot" on page 265
• ":PRINt" on page 130
• ":PRINt?" on page 525

PROBe Commands:
• ":CHANnel<n>:PROBe" on page 177
• ":EXTernal:PROBe" on page 206

PROTection Commands:
• ":CHANnel<n>:PROTection" on page 181
• ":EXTernal:PROTection" on page 209

• Pulse Width (GLITch), ":TRIGger:GLITch Commands" on page 382
• PWIDth, ":MEASure:PWIthd" on page 266

QUALifier Commands:
• ":TRIGger:DURation:QUALifier" on page 358
• ":TRIGger:GLITch:QUALifier" on page 388
• ":TRIGger:IIIC:TRIGger:QUALifier" on page 397

RANGE Commands:
• ":CHANnel<n>:RANGE" on page 182
• ":EXTernal:RANGE" on page 210
• ":FUNCtion:RANGE" on page 218
• ":TIMebase:RANGE" on page 324
• ":TIMebase:WINDow:RANGE" on page 330
• ":TRIGger:DURation:RANGE" on page 359
• ":TRIGger:GLITch:RANGE" on page 389
• "*RCL (Recall)" on page 88
• REFClock, ":TIMebase:REFClock" on page 325

REference Commands:
• ":FUNCtion:REReference" on page 219
• ":TIMebase:REReference" on page 326

REJect, ":TRIGger[:EDGE]:REJect" on page 367

RESet Commands:
• ":SBUS:CAN:COUNt:RESet" on page 300
• ":SBUS:FLEXray:COUNt:RESet" on page 305
• ":TRIGger:SEQuence:RESet" on page 413
• RISetime, ":MEASure:RISetime" on page 267
• "Root (:) Commands" on page 100
• RSIGnal, ":ACQuire:RSIGnal" on page 146
• "*RST (Reset)" on page 89
• ":RUN" on page 131

$ • SAMPlepoint, ":TRIGger:CAN:SAMPlepoint" on page 349
• "*SAV (Save)" on page 92
• ":SBUS:BUSDoctor:ADDRess" on page 294
• ":SBUS:BUSDoctor:BAUDrate" on page 295
• ":SBUS:BUSDoctor:CHANnel" on page 296
• ":SBUS:BUSDoctor:MODE" on page 297
• ":SBUS:CAN:COUNt:ERRor" on page 298
• ":SBUS:CAN:COUNt:OVERload" on page 299
• ":SBUS:CAN:COUNt:RESet" on page 300
• ":SBUS:CAN:COUNt:TOTal" on page 301
• ":SBUS:CAN:COUNt:UTILization" on page 302
• ":SBUS:DISPlay" on page 303
• ":SBUS:FLEXray:COUNt:NULL" on page 304
• ":SBUS:FLEXray:COUNt:RESet" on page 305
• ":SBUS:FLEXray:COUNt:SYNC" on page 306
• ":SBUS:FLEXray:COUNt:TOTal" on page 307
• ":SBUS:IIC:ASIZe" on page 308
• ":SBUS:LIN:PARity" on page 309
• ":SBUS:MODE" on page 310
• ":SBUS:SPI:WIDTh" on page 311
• SCALe Commands:
  • ":CHANnel<n>:SCALE" on page 183
  • ":FUNCtion:SCALE" on page 220
  • ":TIMebase:SCALE" on page 327
  • ":TIMebase:WINDow:SCALE" on page 331
• SCRatch, ":MEASure:SCRatch" on page 512
• SDEViation, ":MEASure:SDEViation" on page 268
• ":SERial" on page 132
• SEGment, ":TRIGger:FLEXray:TIME:SEGment" on page 379
• SETup, ":SYSTem:SETup" on page 317
• SEQuence, ":TRIGger:SEQuence Commands" on page 408
• SHOW, ":MEASure:SHOW" on page 269
• SLOT, ":TRIGger:FLEXray:TIME:SLOT" on page 380
• SIGNal Commands:
  • ":TRIGger:CAN:SIGNal:BAUDrate" on page 350
  • ":TRIGger:CAN:SIGNal:DEFinition" on page 529
  • ":TRIGger:LIN:SIGNal:BAUDrate" on page 403
  • ":TRIGger:LIN:SIGNal:DEFinition" on page 530
• ":SINGle" on page 133
• SIZE Commands:
  • ":DIGital<n>:SIZE" on page 191
  • ":POD<n>:SIZE" on page 289
• SKEW, ":CHANnel<n>:PROBe:SKEW" on page 179
• SLOPe Commands:
  • ":TRIGger:EBURst:SLOPe" on page 363
  • ":TRIGger[:EDGE]:SLOPe" on page 368
  • ":TRIGger:SPI:CLOCk:SLOPe" on page 417
• SOURce Commands:
  • ":DISPlay:SOURce" on page 201
  • ":FUNCTION:SOURce" on page 221
  • ":MEASure:SOURce" on page 270
  • ":TRIGger:CAN:SOURce" on page 351
  • ":TRIGger:GLITch:SOURce" on page 390
  • ":TRIGger:IIC:SOURce:CLOCk" on page 395
  • ":TRIGger:IIC:SOURce:DATA" on page 396
  • ":TRIGger:LIN:SOURce" on page 404
  • ":TRIGger:SPI:SOURce:CLOCk" on page 422
  • ":TRIGger:SPI:SOURce:DATA" on page 423
  • ":TRIGger:SPI:SOURce:FRAME" on page 424
  • ":TRIGger:TV:SOURce" on page 429
  • ":TRIGger:USB:SOURce:DMINus" on page 432
  • ":TRIGger:USB:SOURce:DPLUS" on page 433
  • ":WAVeform:SOURce" on page 456
• SPAN, ".:FUNCTION:SPAN" on page 222
• SPEed, ".:TRIGger:USB:SPEed" on page 434
• SPI Commands:
  • ".:SBUS:SPI:WIDTH" on page 311
  • ".:TRIGger:SPI Commands" on page 416
• SRATe, ".:ACQuire:SRATe" on page 147
• "*:SRE (Service Request Enable)" on page 93
• STANdard, ".:TRIGger:TV:STANdard" on page 430
• START, ".:CALibrate:STARt" on page 162
• STATus Commands:
  • ":CALibrate:STATus" on page 163
  • ".:STATus" on page 134
• "*:STB (Read Status Byte)" on page 95
• ":STOP" on page 135
• SWEep, ".:TRIGger:SWEep" on page 342
• SWITch, ".:CALibrate:SWITch" on page 164
• SYNC, ".:SBUS:FLEXray:COUNt:SYNC" on page 306
• ":SYSTem:DATE" on page 313
• ":SYSTem:DSP" on page 314
• ":SYSTem:ERRor" on page 315
• ":SYSTem:LOCK" on page 316
• ":SYSTem:SETup" on page 317
• ":SYSTem:TIME" on page 319
• TDELta, ":MEASure:TDELta" on page 513
• TEDGe, ":MEASure:TEDGe" on page 272
• TEMPerature, ".:CALibrate:TEMPerature" on page 165
• ":TER (Trigger Event Register)" on page 136
• THReshold Commands:
  • ":CHANnel:THReshold" on page 498
  • ":DIGital<n>:THReshold" on page 192
  • ":MEASure:THResholds" on page 514
  • ":POD<n>:THReshold" on page 290
  • ":TRIGger:THReshold" on page 531
• THResholds, ":MEASure:THResholds" on page 514
• TIME Commands:
• ":CALibrate:TIME" on page 166
• ":SYStem:TIME" on page 319
• ":TRIGger:FLEXray:TIME:CBASe" on page 377
• ":TRIGger:FLEXray:TIME:CREPetition" on page 378
• ":TRIGger:FLEXray:TIME:SEGment" on page 379
• ":TRIGger:FLEXray:TIME:SLOT" on page 380
• ":TIMebase:DELay" on page 527
• ":TIMebase:MODE" on page 322
• ":TIMebase:POSition" on page 323
• ":TIMebase:RANGe" on page 324
• ":TIMebase:REFClock" on page 325
• ":TIMebase:REference" on page 326
• ":TIMebase:SCALE" on page 327
• ":TIMebase:VERNier" on page 328
• ":TIMebase:WINDow:POSition" on page 329
• ":TIMebase:WINDow:RANGe" on page 330
• ":TIMebase:WINDow:SCALE" on page 331
• TIMeout, ":TRIGger:SPI:CLOCk:TIMeout" on page 418
• TIMer, ":TRIGger:SEQuence:TIMer" on page 414
• TMAX, ":MEASure:TMAX" on page 515
• TMIN, ":MEASure:TMIN" on page 516
• TOTal Commands:
  • ":SBUS:CAN:COUNt:TOTal" on page 301
  • ":SBUS:FLEXray:COUNt:TOTal" on page 307
• ":TR (Trigger)" on page 97
• TRIGger Commands:
  • ":TRIGger:CAN:TRIGger" on page 352
  • ":TRIGger:FLEXray:TRIGger" on page 381
  • ":TRIGger:IIC:TRIGger:QUALifier" on page 397
  • ":TRIGger:IIC:TRIGger[:TYPE]" on page 398
  • ":TRIGger:LIN:TRIGger" on page 407
  • ":TRIGger:SEQuence:TRIGger" on page 415
  • ":TRIGger:USB:TRIGger" on page 435
• ":TRIGger:HFReject" on page 336
• ":TRIGger:HOLDoff" on page 337
- ".TRIGger:MODE" on page 338
- ".TRIGger:NREJect" on page 339
- ".TRIGger:PAFFTer" on page 340
- ".TRIGger:SWEep" on page 342
- ".TRIGger:THReshold" on page 531
- ".TRIGger:CAN:ACKNowledge" on page 528
- ".TRIGger:CAN:PAFFTer:DATA" on page 345
- ".TRIGger:CAN:PAFFTer:DATA:LENGth" on page 346
- ".TRIGger:CAN:PAFFTer:ID" on page 347
- ".TRIGger:CAN:PAFFTer:ID:MODE" on page 348
- ".TRIGger:CAN:SAMPLEpoint" on page 349
- ".TRIGger:CAN:SIGNal:BAUDrate" on page 350
- ".TRIGger:CAN:SIGNal:DEFinition" on page 529
- ".TRIGger:CAN:SOURce" on page 351
- ".TRIGger:CAN:TRIGger" on page 352
- ".TRIGger:DURation:GREaterthan" on page 355
- ".TRIGger:DURation:LESSthan" on page 356
- ".TRIGger:DURation:PAFFTer" on page 357
- ".TRIGger:DURation:QUALifier" on page 358
- ".TRIGger:DURation:RANGe" on page 359
- ".TRIGger[:EDGE]:COUpling" on page 365
- ".TRIGger[:EDGE]:LEVel" on page 366
- ".TRIGger[:EDGE]:REJect" on page 367
- ".TRIGger[:EDGE]:SLOPe" on page 368
- ".TRIGger[:EDGE]:SOURce" on page 369
- ".TRIGger:FLEXray:ERRor:TYPE" on page 371
- ".TRIGger:FLEXray:FRAME:CCBase" on page 373
- ".TRIGger:FLEXray:FRAME:CCRepetition" on page 374
- ".TRIGger:FLEXray:FRAME:ID" on page 375
- ".TRIGger:FLEXray:FRAME:TYPE" on page 376
- ".TRIGger:FLEXray:TIME:CBASe" on page 377
- ".TRIGger:FLEXray:TIME:CREPetition" on page 378
- ".TRIGger:FLEXray:TIME:SEGMen" on page 379
- ".TRIGger:FLEXray:TIME:SLOT" on page 380
- ".TRIGger:FLEXray:TRIGger" on page 381
• ".TRIGger:GLITch:GREaterthan" on page 384
• ".TRIGger:GLITch:LESSthan" on page 385
• ".TRIGger:GLITch:LEVel" on page 386
• ".TRIGger:GLITch:POLarity" on page 387
• ".TRIGger:GLITch:QUALifier" on page 388
• ".TRIGger:GLITch:RANGe" on page 389
• ".TRIGger:GLITch:SOURce" on page 390
• ".TRIGger:HFReject" on page 336
• ".TRIGger:HOLDoff" on page 337
• ".TRIGger:IIC:PATTern:ADDRess" on page 392
• ".TRIGger:IIC:PATTern:DATA" on page 393
• ".TRIGger:IIC:PATTern:DATa2" on page 394
• ".TRIGger:IIC:SOURce:CLOCK" on page 395
• ".TRIGger:IIC:SOURce:DATA" on page 396
• ".TRIGger:IIC:TRIGger:QUALifier" on page 397
• ".TRIGger:IIC:TRIGger[TYPE]" on page 398
• ".TRIGger:LIN:SIGNal:BAUDrate" on page 403
• ".TRIGger:LIN:SIGNal:DEFinition" on page 530
• ".TRIGger:LIN:SOURce" on page 404
• ".TRIGger:LIN:TRIGger" on page 407
• ".TRIGger:MODE" on page 338
• ".TRIGger:NREJect" on page 339
• ".TRIGger:PATTern" on page 340
• ".TRIGger:SEQUence:COUNt" on page 409
• ".TRIGger:SEQUence:EDGE" on page 410
• ".TRIGger:SEQUence:FIND" on page 411
• ".TRIGger:SEQUence:PATTern" on page 412
• ".TRIGger:SEQUence:RESet" on page 413
• ".TRIGger:SEQUence:TIMer" on page 414
• ".TRIGger:SEQUence:TRIGger" on page 415
• ".TRIGger:SPI:CLOCk:SLOPe" on page 417
• ".TRIGger:SPI:CLOCk:TIMEout" on page 418
• ".TRIGger:SPI:FRAMing" on page 419
• ".TRIGger:SPI:PATTern:DATA" on page 420
• ".TRIGger:SPI:PATTern:WIDTH" on page 421
- ".:TRIGger:SPI:SOURce:CLOCk" on page 422
- ":TRIGger:SPI:SOURce:DATA" on page 423
- ":TRIGger:SPI:SOURce:FRAMe" on page 424
- ":TRIGger:SWEep*" on page 342
- ":TRIGger:THReshold" on page 531
- ":TRIGger:TV:LINE" on page 426
- ":TRIGger:TV:MODE*" on page 427
- ":TRIGger:TV:POLarity*" on page 428
- ":TRIGger:TV:SOURce*" on page 429
- ":TRIGger:TV:STANdard*" on page 430
- ":TRIGger:TV:TVMode*" on page 532
- ":TRIGger:USB:SOURce:DMINus" on page 432
- ":TRIGger:USB:SOURce:DPLus" on page 433
- ":TRIGger:USB:SPEed" on page 434
- ":TRIGger:USB:TRIGger" on page 435
- "*TST (Self Test)" on page 98
- TSTArt, ":MEASure:TSTArt" on page 517
- TSTOP, ":MEASure:TSTOP" on page 518
- TV, ":TRIGger:TV Commands" on page 425
- TVALue, ":MEASure:TV ALue" on page 274
- TVOLt, ":MEASure:TVOLt" on page 519
- TYPE Commands:
  - ":ACQuire:TYPE" on page 148
  - ":WAVeform:TYPE" on page 460
  - ":TRIGger:FLEXray:ERRor:TYPE" on page 371
  - ":TRIGger:FLEXray:FRAMe:TYPE" on page 376
  - ":TRIGger:IIC:TRIGger[:TYPE]" on page 398
- UNITs Commands:
  - ":CHANnel<n>:UNITs" on page 184
  - ":EXTernal:UNITs" on page 211
  - UNSigned, ":WAVEform:UNSigned" on page 461
  - UPPer, ":MEASure:UPPer" on page 521
  - USB, ":TRIGger:USB Commands" on page 431
  - UTILization, ":SBUS:CAN:COUNt:UTILization" on page 302
V

- VAMPlitude, ":MEASure:VAMPlitude" on page 276
- VAverage, ":MEASure:VAverage" on page 277
- VBASe, ":MEASure:VBASe" on page 278
- VDELta, ":MEASure:VDELta" on page 522
- VECTors, ":DISPlay:VECTors" on page 202
- VERNier, ":CHANnel<n>:VERNier" on page 185
- ":VIEW" on page 137
- VMAX, ":MEASure:VMAX" on page 279
- VMIN, ":MEASure:VMIN" on page 280
- VPP, ":MEASure:VPP" on page 281
- VRMS, ":MEASure:VRMS" on page 282
- VSTArt, ":MEASure:VSTArt" on page 523
- VSTOp, ":MEASure:VSTOp" on page 524
- VTIME, ":MEASure:VTIME" on page 283
- VTOP, ":MEASure:VTOP" on page 284

W

- "*WAI (Wait To Continue)" on page 99
- ":WAVEform:BYTeorder" on page 444
- ":WAVEform:COUNT" on page 445
- ":WAVEform:DATA" on page 446
- ":WAVEform:FORMat" on page 448
- ":WAVEform:POINts" on page 449
- ":WAVEform:POINts:MODE" on page 451
- ":WAVEform:PREamble" on page 453
- ":WAVEform:SOURce" on page 456
- ":WAVEform:TYPE" on page 460
- ":WAVEform:UNSIGNED" on page 461
- ":WAVEform:VIEW" on page 462
- ":WAVEform:XINCrement" on page 463
- ":WAVEform:XORigin" on page 464
- ":WAVEform:XREFerence" on page 465
- ":WAVEform:YINCrement" on page 466
- ":WAVEform:YORigin" on page 467
- ":WAVEform:YREFerence" on page 468

WIDTh Commands:
  - ":SBUS:SPI:WIDTh" on page 311
4 Commands A-Z

- ":TRIGger:SPI:PATrern:WIDTh" on page 421
- WINDow, ":FUNCTION:WINDow" on page 223
  
  X  
  - X1Position, ":MARKer:X1Position" on page 235
  - X1Y1source, ":MARKer:X1Y1source" on page 236
  - X2Position, ":MARKer:X2Position" on page 237
  - X2Y2source, ":MARKer:X2Y2source" on page 238
  - XDELta, ":MARKer:XDELta" on page 239
  - XINCrement, ":WAVeform:XINCrement" on page 463
  - XMAX, ":MEASure:XMAX" on page 285
  - XMIN, ":MEASure:XMIN" on page 286
  - XORigin, ":WAVeform:XORigin" on page 464
  - XREFerence, ":WAVeform:XREFerence" on page 465

  Y  
  - Y1Position, ":MARKer:Y1Position" on page 240
  - Y2Position, ":MARKer:Y2Position" on page 241
  - YDELta, ":MARKer:YDELta" on page 242
  - YINCrement, ":WAVeform:YINCrement" on page 466
  - YORigin, ":WAVeform:YORigin" on page 467
  - YREFerence, ":WAVeform:YREFerence" on page 468
Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs (see “Obsolete Commands” on page 564).

<table>
<thead>
<tr>
<th>Obsolete Command</th>
<th>Current Command Equivalent</th>
<th>Behavior Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANALog&lt;n&gt;:BWLimit</td>
<td>:CHANnel&lt;n&gt;:BWLimit (see page 170)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:COUplng</td>
<td>:CHANnel&lt;n&gt;:COUplng (see page 171)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:INVert</td>
<td>:CHANnel&lt;n&gt;:INVert (see page 174)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:LABel</td>
<td>:CHANnel&lt;n&gt;:LABel (see page 175)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:OFFSet</td>
<td>:CHANnel&lt;n&gt;:OFFSet (see page 176)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:PROBe</td>
<td>:CHANnel&lt;n&gt;:PROBe (see page 177)</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:PMODE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>ANALog&lt;n&gt;:RANGe</td>
<td>:CHANnel&lt;n&gt;:RANGe (see page 182)</td>
<td></td>
</tr>
<tr>
<td>:CHANnel:ACTivity (see page 496)</td>
<td>:ACTivity (see page 103)</td>
<td></td>
</tr>
<tr>
<td>:CHANnel:LABel (see page 497)</td>
<td>:CHANnel&lt;n&gt;:LABel (see page 175) or :DIGital&lt;n&gt;:LABel (see page 189)</td>
<td>use CHANnel&lt;n&gt;:LABel for analog channels and use DIGital&lt;n&gt;:LABel for digital channels</td>
</tr>
<tr>
<td>:CHANnel:THReshold (see page 498)</td>
<td>:POD&lt;n&gt;:THReshold (see page 290) or :DIGital&lt;n&gt;:THReshold (see page 192)</td>
<td></td>
</tr>
<tr>
<td>:CHANnel2:SKEW (see page 499)</td>
<td>:CHANnel&lt;n&gt;:PROBe:SKEW (see page 179)</td>
<td></td>
</tr>
</tbody>
</table>
### Obsolete and Discontinued Commands

<table>
<thead>
<tr>
<th>Obsolete Command</th>
<th>Current Command Equivalent</th>
<th>Behavior Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:CHANnel&lt;n&gt;:INPut (see page 500)</td>
<td>:CHANnel&lt;n&gt;:IMPedance (see page 173)</td>
<td></td>
</tr>
<tr>
<td>:CHANnel&lt;n&gt;:PMODe (see page 501)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:DISPlay:CONNect (see page 502)</td>
<td>:DISPlay:VECTors (see page 202)</td>
<td></td>
</tr>
<tr>
<td>:DISPlay:ORDer (see page 503)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:ERASE (see page 504)</td>
<td>:CDSplay (see page 110)</td>
<td></td>
</tr>
<tr>
<td>:EXTERNal:INPut (see page 505)</td>
<td>:EXTERNal:IMPedance (see page 205)</td>
<td></td>
</tr>
<tr>
<td>:EXTERNal:PMODe (see page 506)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>FUNCTION1, FUNCTION2</td>
<td>FUNCTION Commands (see page 212)</td>
<td>ADD not included</td>
</tr>
<tr>
<td>:FUNCTION:VIEW (see page 507)</td>
<td>:FUNCTION:DISPLAY (see page 215)</td>
<td></td>
</tr>
<tr>
<td>:HARDcopy:DESTination (see page 508)</td>
<td>:HARDcopy:FILename (see page 227)</td>
<td></td>
</tr>
<tr>
<td>:HARDcopy:DEVice (see page 509)</td>
<td>:HARDcopy:FORMat (see page 228)</td>
<td>PLOTter, THINKjet not supported; TIF, BMP, CSV, SEiko added</td>
</tr>
<tr>
<td>:HARDcopy:GRAYscale (see page 510)</td>
<td>:HARDcopy:PALETTE (see page 230)</td>
<td></td>
</tr>
<tr>
<td>:MEASURE:LOWer (see page 511)</td>
<td>:MEASURE:DEFine:THResholds (see page 252)</td>
<td>MEASURE:DEFine:THResholds can define absolute values or percentage</td>
</tr>
<tr>
<td>:MEASURE:SCRatch (see page 512)</td>
<td>:MEASURE:CLEAR (see page 250)</td>
<td></td>
</tr>
<tr>
<td>:MEASURE:TDELta (see page 513)</td>
<td>:MARKer:XDELta (see page 239)</td>
<td></td>
</tr>
<tr>
<td>:MEASURE:THResholds (see page 514)</td>
<td>:MEASURE:DEFine:THResholds (see page 252)</td>
<td>MEASURE:DEFine:THResholds can define absolute values or percentage</td>
</tr>
<tr>
<td>:MEASURE:TMAX (see page 515)</td>
<td>:MEASURE:XMAX (see page 285)</td>
<td></td>
</tr>
<tr>
<td>:MEASURE:TMIN (see page 516)</td>
<td>:MEASURE:XMIN (see page 286)</td>
<td></td>
</tr>
<tr>
<td>:MEASURE:TSTArt (see page 517)</td>
<td>:MARKer:X1Position (see page 235)</td>
<td></td>
</tr>
</tbody>
</table>
### Obsolete and Discontinued Commands

**Discontinued commands** are commands that were used by previous oscilloscopes, but are not supported by the 6000 Series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

<table>
<thead>
<tr>
<th>Obsolete Command</th>
<th>Current Command Equivalent</th>
<th>Behavior Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>:MEASure:TSTOp (see page 518)</td>
<td>:MARKer:X2Position (see page 237)</td>
<td></td>
</tr>
<tr>
<td>:MEASure:TVOLT (see page 519)</td>
<td>:MEASure:TVALue (see page 274)</td>
<td>TVALue measures additional values such as db, Vs, etc.</td>
</tr>
<tr>
<td>:MEASure:UPPer (see page 521)</td>
<td>:MEASure:DEFine:THResholds (see page 252)</td>
<td>MEASure:DEFine:THResholds can define absolute values or percentage</td>
</tr>
<tr>
<td>:MEASure:VDELta (see page 522)</td>
<td>:MARKer:YDELta (see page 242)</td>
<td></td>
</tr>
<tr>
<td>:MEASure:VSTArt (see page 523)</td>
<td>:MARKer:Y1Position (see page 240)</td>
<td></td>
</tr>
<tr>
<td>:MEASure:VSTOp (see page 524)</td>
<td>:MARKer:Y2Position (see page 241)</td>
<td></td>
</tr>
<tr>
<td>:PRINt? (see page 525)</td>
<td>:DISPlay:DATA? (see page 196)</td>
<td></td>
</tr>
<tr>
<td>:TIMebase:DELay (see page 527)</td>
<td>:TIMebase:POSition (see page 323) or :TIMebase:WINDow:POSition (see page 329)</td>
<td>TIMebase:POSition is position value of main time base; TIMebase:WINDow:POSition is position value of delayed time base window.</td>
</tr>
<tr>
<td>:TRIGger:CAN:ACKNowledge (see page 528)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:CAN:SIGNal:DEFinition (see page 529)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:LIN:SIGNal:DEFinition (see page 530)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:THReshold (see page 531)</td>
<td>:POD&lt;n&gt;:THReshold (see page 290) or :DIGital&lt;n&gt;:THReshold (see page 192)</td>
<td></td>
</tr>
<tr>
<td>:TRIGger:TV:TVMode (see page 532)</td>
<td>:TRIGger:TV:MODE (see page 427)</td>
<td></td>
</tr>
</tbody>
</table>
### Obsolete and Discontinued Commands

<table>
<thead>
<tr>
<th>Discontinued Command</th>
<th>Current Command Equivalent</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTore</td>
<td>:DISPlay:PERSistance INFinite (see page 200)</td>
<td></td>
</tr>
<tr>
<td>CHANnel:MATH</td>
<td>:FUNCtion:OPERation (see page 217)</td>
<td>ADD not included</td>
</tr>
<tr>
<td>CHANnel&lt;n&gt;:PROTect</td>
<td>:CHANnel&lt;n&gt;:PROtection (see page 181)</td>
<td>Previous form of this command was used to enable/disable 50Ω protection. The new command resets a tripped protect and the query returns the status of TRIPed or NORMal.</td>
</tr>
<tr>
<td>DISPlay:INVerse</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:COLumn</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:GRID</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:LINE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:PIXel</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:POSition</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:ROW</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DISPlay:TEXT</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>FUNCTION:MOVE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>FUNCTION:PEAKs</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>HARDcopy:ADDRess</td>
<td>none</td>
<td>Only parallel printer port is supported. GPIB printing not supported</td>
</tr>
<tr>
<td>MASK</td>
<td>none</td>
<td>All commands discontinued, feature not available</td>
</tr>
<tr>
<td>SYSTem:KEY</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>TEST:ALL</td>
<td>*TST (Self Test) (see page 98)</td>
<td></td>
</tr>
<tr>
<td>TRACE subsystem</td>
<td>none</td>
<td>All commands discontinued, feature not available</td>
</tr>
<tr>
<td>TRIGger:ADVanced subsystem</td>
<td></td>
<td>Use new GLITch, PATTern, or TV trigger modes</td>
</tr>
<tr>
<td>TRIGger:TV:FIELD</td>
<td>:TRIGger:TV:MODE (see page 427)</td>
<td></td>
</tr>
<tr>
<td>TRIGger:TV:VIR</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>VAUToscale</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>
Discontinued Parameters

Some previous oscilloscope queries returned control setting values of OFF and ON. The 6000 Series oscilloscopes only return the enumerated values 0 (for off) and 1 (for on).
**:CHANnel:ACTivity**

(see page 564)

**Command Syntax**

:CHANnel:ACTivity

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

**NOTE**

The :CHANnel:ACTivity command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :ACTivity command (see page 103) instead.

**Query Syntax**

:CHANnel:ACTivity?

The :CHANnel:ACTivity? query returns the active edges since the last clear, and returns the current logic levels.

**Return Format**

<edges>,<levels><NL>

<edges> ::= presence of edges (32-bit integer in NR1 format).

<levels> ::= logical highs or lows (32-bit integer in NR1 format).

**NOTE**

A bit equal to zero indicates that no edges were detected at the specified threshold since the last clear on that channel. Edges may have occurred that were not detected because of the threshold setting.

A bit equal to one indicates that edges have been detected at the specified threshold since the last clear on that channel.
**:CHANnel:LABel**

(see page 564)

**Command Syntax**

:CHANnel:LABel <source_text><string>

<source_text> ::= {CHANnel1 | CHANnel2 | DIGital0,...,DIGital15}

<string> ::= quoted ASCII string

The :CHANnel:LABel command sets the source text to the string that follows. Setting a channel will also result in the name being added to the label list.

**NOTE**

The :CHANnel:LABel command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:LABel command (see page 175) or :DIGital<n>:LABel command (see page 189) for the 6000 Series oscilloscopes.

**Query Syntax**

:CHANnel:LABel?

The :CHANnel:LABel? query returns the label associated with a particular analog channel.

**Return Format**

<string><NL>

<string> ::= quoted ASCII string
### :CHANnel:THReshold

(see page 564)

**Command Syntax**

:CHANnel:THReshold <channel group>, <threshold type> [, <value>]

- `<channel group>` ::= {POD1 | POD2}
- `<threshold type>` ::= {CMOS | ECL | TTL | USERdef}
- `<value>` ::= voltage for USERdef in NR3 format [volt_type]

[volt_type] ::= {V | mV (-3) | uV (-6)}

The :CHANnel:THReshold command sets the threshold for a group of channels. The threshold is either set to a predefined value or to a user-defined value. For the predefined value, the voltage parameter is ignored.

**NOTE**
The :CHANnel:THReshold command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :POD<n>:THReshold command (see page 290) or :DIGital<n>:THReshold command (see page 192) for the 6000 Series oscilloscopes.

**Query Syntax**

:CHANnel:THReshold? <channel group>

The :CHANnel:THReshold? query returns the voltage and threshold text for a specific group of channels.

**Return Format**

<threshold type> [, <value>]<NL>

- `<threshold type>` ::= {CMOS | ECL | TTL | USERdef}
- `<value>` ::= voltage for USERdef (float 32 NR3)

**NOTE**
- CMOS = 2.5V
- TTL = 1.5V
- ECL = -1.3V
- USERdef ::= -6.0V to 6.0V
:**CHANnel2:SKEW**

(see page 564)

**Command Syntax**

:CHANnel2:SKEW <skew value>

<skew value> ::= skew time in NR3 format

<skew value> ::= -100 ns to +100 ns

The :CHANnel2:SKEW command sets the skew between channels 1 and 2. The maximum skew is +/- 100 ns. You can use the oscilloscope's analog probe skew control to remove cable delay errors between channel 1 and channel 2.

**NOTE**

The :CHANnel2:SKEW command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:PROBe:SKEW command (see page 179) instead.

**NOTE**

This command is only valid for the two channel oscilloscope models.

**Query Syntax**

:CHANnel2:SKEW?

The :CHANnel2:SKEW? query returns the current probe skew setting for the selected channel.

**Return Format**

<skew value><NL>

<skew value> ::= skew value in NR3 format

**See Also**

- "Introduction to :CHANnel<n> Commands" on page 168
5 Obsolete and Discontinued Commands

:CHANnel\langle n\rangle:INPut

(see page 564)

Command Syntax

:CHANnel\langle n\rangle:INPut \langle\text{impedance}\rangle

\langle\text{impedance}\rangle ::= \{\text{ONEMeg} \mid \text{FIFTy}\}

\langle n\rangle ::= \{1 \mid 2 \mid 3 \mid 4\} \text{ for the four channel oscilloscope models}

\langle n\rangle ::= \{1 \mid 2\} \text{ for the two channel oscilloscope models}

The :CHANnel\langle n\rangle:INPut command selects the input impedance setting for the specified channel. The legal values for this command are ONEMeg (1 M\(\Omega\)) and FIFTy (50\(\Omega\)).

NOTE

The :CHANnel\langle n\rangle:INPut command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel\langle n\rangle:IMPedance command (see page 173) instead.

Query Syntax

:CHANnel\langle n\rangle:INPut?

The :CHANnel\langle n\rangle:INPut? query returns the current input impedance setting for the specified channel.

Return Format

\langle\text{impedance value}\rangle\langle\text{NL}\rangle

\langle\text{impedance value}\rangle ::= \{\text{ONEM} \mid \text{FIFT}\}
:CHANnel<n>:PMODe

(see page 564)

Command Syntax

CHANnel<n>:PMODe <pmode value>

<pmode value> ::= (AUTo | MANual)

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models
<n> ::= {1 | 2} for the two channel oscilloscope models

The probe sense mode is controlled internally and cannot be set. If a probe with sense is connected to the specified channel, auto sensing is enabled; otherwise, the mode is manual.

If the PMODe sent matches the oscilloscope's setting, the command will be accepted. Otherwise, a setting conflict error is generated.

NOTE

The :CHANnel<n>:PMODe command is an obsolete command provided for compatibility to previous oscilloscopes.

Query Syntax

CHANnel<n>:PMODe?

The :CHANnel<n>:PMODe? query returns AUT if an autosense probe is attached and MAN otherwise.

Return Format

<pmode value><NL>

<pmode value> ::= (AUT | MAN)
## :DISPlay:CONNect

(see page 564)

**Command Syntax**

:DISPlay:CONNect <connect>

<connect> ::= {{ 1 | ON} | {0 | OFF}}

The :DISPlay:CONNect command turns vectors on and off. When vectors are turned on, the oscilloscope displays lines connecting sampled data points. When vectors are turned off, only the sampled data is displayed.

### NOTE

The :DISPlay:CONNect command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISPlay:VECTors command (see page 202) instead.

**Query Syntax**

:DISPlay:CONNect?

The :DISPlay:CONNect? query returns the current state of the vectors setting.

**Return Format**

<connect><NL>

<connect> ::= {1 | 0}

**See Also**

- ":DISPlay:VECTors" on page 202
:DISPLAY:ORDer

(see page 564)

**Query Syntax**

:DISPLAY:ORDer?

The :DISPLAY:ORDer? query returns a list of digital channel numbers in screen order, from top to bottom, separated by commas. Busing is displayed as digital channels with no separator. For example, in the following list, the bus consists of digital channels 4 and 5: DIG1, DIG4 DIG5, DIG7.

**NOTE**

The :DISPLAY:ORDer command is an obsolete command provided for compatibility to previous oscilloscopes. This command is only available on the MSO models.

**Return Format**

<order><NL>

<order> ::= Unquoted ASCII string

**NOTE**

A return value is included for each digital channel. A return value of NONE indicates that a channel is turned off.

**See Also**

- "::DIGital<n>::POSition" on page 190

**Example Code**

' DISP_ORDER - Set the order the channels are displayed on the analyzer. You can enter between 1 and 32 channels at one time. If you leave out channels, they will not be displayed.

' Display ONLY channel 0 and channel 10 in that order. myScope.WriteString "::DISPLAY:ORDER 0,10"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
5 Obsolete and Discontinued Commands

:ERASe

(see page 564)

Command Syntax

:ERASe

The :ERASe command erases the screen.

NOTE

The :ERASe command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISplay command (see page 110) instead.
**:EXTernal:INPut**

(see page 564)

**Command Syntax**

`:EXTernal:INPut <impedance>`

<impedance> ::= {ONEMeg | FIFTy}

The :EXTernal:INPut command selects the input impedance setting for the external trigger. The legal values for this command are ONEMeg (1 MΩ) and FIFTy (50Ω).

**NOTE**

The :EXTernal:INPut command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :EXTernal:IMPedance command (see page 205) instead.

**Query Syntax**

`:EXTernal:INPut?`

The :EXTernal:INPut? query returns the current input impedance setting for the external trigger.

**Return Format**

<impedance value><NL>

<impedance value> ::= {ONE | FIFT}

**See Also**

- "Introduction to :EXTernal Trigger Commands" on page 203
- "Introduction to :TRIGger Commands" on page 332
- ":CHANnel<n>:IMPedance" on page 173
5 Obsolete and Discontinued Commands

:EXTernal:PMODE

(see page 564)

Command Syntax

:EXTernal:PMODE <pmode value>

<pmode value> ::= {AUTo | MANual}

The probe sense mode is controlled internally and cannot be set. If a probe with sense is connected to the specified channel, auto sensing is enabled; otherwise, the mode is manual.

If the pmode sent matches the oscilloscope's setting, the command will be accepted. Otherwise, a setting conflict error is generated.

NOTE

The :EXTernal:PMODE command is an obsolete command provided for compatibility to previous oscilloscopes.

Query Syntax

:EXTernal:PMODE?

The :EXTernal:PMODE? query returns AUT if an autosense probe is attached and MAN otherwise.

Return Format

<pmode value><NL>

<pmode value> ::= {AUT | MAN}
**:FUNCTION:VIEW**

(see page 564)

**Command Syntax**

:FUNCTION:VIEW <view>

<view> ::= {{1 | ON} | {0 | OFF}}

The :FUNCTION:VIEW command turns the selected function on or off. When ON is selected, the function performs as specified using the other FUNCTION commands. When OFF is selected, function is neither calculated nor displayed.

**NOTE**

The :FUNCTION:VIEW command is provided for backward compatibility to previous oscilloscopes. Use the :FUNCTION:DISPLAY command (see page 215) instead.

**Query Syntax**

:FUNCTION:VIEW?

The :FUNCTION:VIEW? query returns the current state of the selected function.

**Return Format**

<view><NL>

<view> ::= {1 | 0}
### :HARDcopy:DESTination

(see page 564)

#### Command Syntax

\[
\text{:HARDcopy:DESTination} \ <\text{destination}>
\]

\[
<\text{destination}> ::= \{\text{CENTronics} \mid \text{FLOppy}\}
\]

The :HARDcopy:DESTination command sets the hardcopy destination.

#### Query Syntax

\[
\text{:HARDcopy:DESTination}\,?\]

The :HARDcopy:DESTination? query returns the selected hardcopy destination.

#### Return Format

\[
<\text{destination}><\text{NL}>
\]

\[
<\text{destination}> ::= \{\text{CENT} \mid \text{FLOp}\}
\]

#### See Also

- "Introduction to :HARDcopy Commands" on page 224
- ".:HARDcopy:FORMat" on page 228
**:HARDcopy:DEVice**

(see page 564)

**Command Syntax**

`:HARDcopy:DEVice <device>`

`<device> ::= {TIFF | GIF | BMP | LASerjet | EPSoN | DESKjet | BWDeskjet | SEIKo}`

The :HARDcopy:DEVice command sets the hardcopy device type.

### NOTE

BWDeskjet option refers to the monochrome Deskjet printer.

### NOTE

The :HARDcopy:DEVice command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:FORMat command (see page 228) instead.

**Query Syntax**

`:HARDcopy:DEVice?`

The :HARDcopy:DEVice? query returns the selected hardcopy device type.

**Return Format**

`<device><NL>`

`<device> ::= {TIFF | GIF | BMP | LAS | EPS | DESK | BWD | SEIK}`
510 Agilent 6000 Series Oscilloscopes Programmer’s Reference

5 Obsolete and Discontinued Commands

:HELLDcopy:GRAYscale

0 (see page 564)

Command Syntax

:HELLDcopy:GRAYscale <gray>

<gray> ::= {{OFF | 0} | {ON | 1}}

The :HELLDcopy:GRAYscale command controls whether grayscaling is performed in the hardcopy dump.

Query Syntax

:HELLDcopy:GRAYscale?

The :HELLDcopy:GRAYscale? query returns a flag indicating whether grayscaling is performed in the hardcopy dump.

Return Format

<gray><NL>

<gray> ::= {0 | 1}

See Also

• "Introduction to :HELLDcopy Commands" on page 224

NOTE

The :HELLDcopy:GRAYscale command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HELLDcopy:PALETTE command (see page 230) instead. (".:HELLDcopy:GRAYscale ON" is the same as ":HELLDcopy:PALETTE GRAYscale" and ":HELLDcopy:GRAYscale OFF" is the same as ":HELLDcopy:PALETTE COLOR").
**:MEASure:LOWer**

(see page 564)

**Command Syntax**

:MEASure:LOWer <voltage>

The :MEASure:LOWer command sets the lower measurement threshold value. This value and the UPPer value represent absolute values when the thresholds are ABSolute and percentage when the thresholds are PERCent as defined by the :MEASure:DEFine THResholds command.

**NOTE**

The :MEASure:LOWer command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THResholds command (see page 252) instead.

**Query Syntax**

:MEASure:LOWer?

The :MEASure:LOWer? query returns the current lower threshold level.

**Return Format**

<voltage><NL>

<voltage> ::= the user-defined lower threshold in volts in NR3 format

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:THResholds" on page 514
- ":MEASure:UPPer" on page 521
5 Obsolete and Discontinued Commands

:MEASURE:SCRatch

(see page 564)

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 250) instead.
:MEASure:TDELta

(see page 564)

Query Syntax

:MEASure:TDELta?

The :MEASure:TDELta? query returns the time difference between the Tstop marker (X2 cursor) and the Tstart marker (X1 cursor).

Tdelta = Tstop - Tstart

Tstart is the time at the start marker (X1 cursor) and Tstop is the time at the stop marker (X2 cursor). No measurement is made when the :MEASure:TDELta? query is received by the oscilloscope. The delta time value that is output is the current value. This is the same value as the front-panel cursors delta X value.

NOTE

The :MEASure:TDELta command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:XDELta command (see page 239) instead.

Return Format

<value><NL>

<value> ::= time difference between start and stop markers in NR3 format

See Also

- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- ":MARKer:X1Position" on page 235
- ":MARKer:X2Position" on page 237
- ":MARKer:XDELta" on page 239
- ":MEASure:TSTArt" on page 517
- ":MEASure:TSTOP" on page 518
**Obsolete and Discontinued Commands**

---

**:MEASure:THResholds**

(see page 564)

**Command Syntax**

:MEASure:THResholds {T1090 | T2080 | VOLTage}

The :MEASure:THResholds command selects the thresholds used when making time measurements.

**NOTE**

The :MEASure:THResholds command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THResholds command (see page 252) instead.

---

**Query Syntax**

:MEASure:THResholds?

The :MEASure:THResholds? query returns the current thresholds selected when making time measurements.

**Return Format**

{T1090 | T2080 | VOLTage}<NL>

{T1090} uses the 10% and 90% levels of the selected waveform.

{T2080} uses the 20% and 80% levels of the selected waveform.

{VOLTage} uses the upper and lower voltage thresholds set by the UPPer and LOWer commands on the selected waveform.

**See Also**

- "Introduction to :MEASure Commands" on page 248
- ":MEASure:LOWer" on page 511
- ":MEASure:UPPer" on page 521
:MEASure:TMAX

(see page 564)

Command Syntax
:MEASure:TMAX [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

The :MEASure:TMAX command installs a screen measurement and starts an X-at-Max-Y measurement on the selected waveform. If the optional source is specified, the current source is modified.

Query Syntax
:MEASure:TMAX? [<source>]

The :MEASure:TMAX? query returns the horizontal axis value at which the maximum vertical value occurs on the current source. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

Return Format
<value><NL>

<value> ::= time at maximum in NR3 format

See Also
- "Introduction to :MEASure Commands" on page 248
- ":MEASure:TMIN" on page 516
- ":MEASure:XMAX" on page 285
- ":MEASure:XMIN" on page 286

NOTE

The :MEASure:TMAX command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:XMAX command (see page 285) instead.
5 Obsolete and Discontinued Commands

:MEASure:TMIN

(see page 564)

Command Syntax

:MEASure:TMIN [<source>]

<source> ::= {CHANnel<n> | FUNCtion | MATH}

<n> ::= {1 | 2 | 3 | 4} for the four channel oscilloscope models

The :MEASure:TMIN command installs a screen measurement and starts an X-at-Min-Y measurement on the selected waveform. If the optional source is specified, the current source is modified.

NOTE
The :MEASure:TMIN command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:XMIN command (see page 286) instead.

Query Syntax

:MEASure:TMIN? [<source>]

The :MEASure:TMIN? query returns the horizontal axis value at which the minimum vertical value occurs on the current source. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

Return Format

/value/<NL>

/value> ::= time at minimum in NR3 format

See Also

• "Introduction to :MEASure Commands" on page 248

• ":MEASure:TMAX" on page 515

• ":MEASure:XMAX" on page 285

• ":MEASure:XMIN" on page 286
:MEASure:TSTArt

Command Syntax

```
:MEASure:TSTArt <value> [suffix]
```

<value> ::= time at the start marker in seconds

[suffix] ::= {s | ms | us | ns | ps}

The :MEASure:TSTArt command moves the start marker (X1 cursor) to the specified time with respect to the trigger time.

Query Syntax

```
:MEASure:TSTArt?
```

The :MEASure:TSTArt? query returns the time at the start marker (X1 cursor).

Return Format

```
<value><NL>
```

<value> ::= time at the start marker in NR3 format

NOTE

The short form of this command, TSTA, does not follow the defined Long Form to Short Form Truncation Rules (see page 566). The normal short form "TST" would be the same for both TSTArt and TSTOp, so sending TST for the TSTArt command produces an error.

NOTE

The :MEASure:TSTArt command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:X1Position command (see page 235) instead.

See Also

- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- ".:MARKer:X1Position" on page 235
- ".:MARKer:X2Position" on page 237
- ".:MARKer:XDELta" on page 239
- ".:MEASure:TDELta" on page 513
- ".:MEASure:TSTOp" on page 518
:MEASure:TSTOp

(see page 564)

Command Syntax

:MEASure:TSTOp <value> [suffix]

<value> ::= time at the stop marker in seconds
[suffix] ::= {s | ms | us | ns | ps}

The :MEASure:TSTOp command moves the stop marker (X2 cursor) to the specified time with respect to the trigger time.

NOTE

The short form of this command, TSTO, does not follow the defined Long Form to Short Form Truncation Rules (see page 566). The normal short form "TST" would be the same for both TSTArt and TSTOp, so sending TST for the TSTOp command produces an error.

NOTE

The :MEASure:TSTOp command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:X2Position command (see page 237) instead.

Query Syntax

:MEASure:TSTOp?

The :MEASure:TSTOp? query returns the time at the stop marker (X2 cursor).

Return Format

<value><NL>

<value> ::= time at the stop marker in NR3 format

See Also

- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- "MARKer:X1Position" on page 235
- "MARKer:X2Position" on page 237
- "MARKer:XDELta" on page 239
- "MEASure:TDELta" on page 513
- "MEASure:TSTArt" on page 517
:MEASure:TVOLt

(see page 564)

Query Syntax

:MEASure:TVOLt? <value>, [<slope>][<occurrence>], [<source>]

:value> ::= the voltage level that the waveform must cross.

:slope> ::= direction of the waveform. A rising slope is indicated by a plus sign (+). A falling edge is indicated by a minus sign (-).

:occurrence> ::= the transition to be reported. If the occurrence number is one, the first crossing is reported. If the number is two, the second crossing is reported, etc.

:source> ::= {<digital channels> | CHANnel<n> | FUNCTION | MATH}

When the :MEASure:TVOLt? query is sent, the displayed signal is searched for the specified voltage level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to the query.

The specified voltage can be negative or positive. To specify a negative voltage, use a minus sign (-). The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of the occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the specified voltage level in the positive direction. Once this voltage crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage for the specified number of times in the direction specified.

If the optional source parameter is specified, the current source is modified.

NOTE

The :MEASure:TVOLt command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:TVALue command (see page 274) for the 6000 Series oscilloscopes.

Return Format

:value><NL>
5 Obsolete and Discontinued Commands

\[ \texttt{<value>} ::= \text{time in seconds of the specified voltage crossing}
\]

\[ \text{in NR3 format} \]
:MEASure:UPPer

(see page 564)

Command Syntax

:MEASure:UPPer <value>

The :MEASure:UPPer command sets the upper measurement threshold value. This value and the LOWer value represent absolute values when the thresholds are ABSolute and percentage when the thresholds are PERCent as defined by the :MEASure:DEFine THResholds command.

NOTE

The :MEASure:UPPer command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:DEFine THResholds command (see page 252) instead.

Query Syntax

:MEASure:UPPer?

The :MEASure:UPPer? query returns the current upper threshold level.

Return Format

<value><NL>

<value> ::= the user-defined upper threshold in NR3 format

See Also

- "Introduction to :MEASure Commands" on page 248
- "::MEASure:LOWer" on page 511
- "::MEASure:THResholds" on page 514
### :MEASure:VDELta

(see page 564)

**Query Syntax**

:MEASure:VDELta?

The :MEASure:VDELta? query returns the voltage difference between vertical marker 1 (Y1 cursor) and vertical marker 2 (Y2 cursor). No measurement is made when the :MEASure:VDELta? query is received by the oscilloscope. The delta value that is returned is the current value. This is the same value as the front-panel cursors delta Y value.

\[
\text{VDELta} = \text{value at marker 2} - \text{value at marker 1}
\]

**NOTE**

The :MEASure:VDELta command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:YDELta command (see page 242) instead.

**Return Format**

\[<\text{value}>\text{<NL>}\]

\[<\text{value}> ::= \text{delta V value in NR1 format}\]

**See Also**

- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- ":MARKer:Y1Position" on page 240
- ":MARKer:Y2Position" on page 241
- ":MARKer:YDELta" on page 242
- ":MEASure:TDELta" on page 513
- ":MEASure:TSTArt" on page 517
:MEASure:VSTArt

Command Syntax

:MEASure:VSTArt <vstart_argument>

<vstart_argument> ::= value for vertical marker 1

The :MEASure:VSTArt command moves the vertical marker (Y1 cursor) to the specified value corresponding to the selected source. The source can be selected by the MARKer:X1Y1source command.

Query Syntax

:MEASure:VSTArt?

The :MEASure:VSTArt? query returns the current value of the Y1 cursor.

Return Format

<value><NL>

<value> ::= voltage at voltage marker 1 in NR3 format

See Also

- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- ":MARKer:Y1Position" on page 240
- ":MARKer:Y2Position" on page 241
- ":MARKer:YDELta" on page 242
- ":MARKer:X1Y1source" on page 236
- ":MEASure:SOURce" on page 270
- ":MEASure:TDELta" on page 513
- ":MEASure:TSTArt" on page 517

NOTE

The short form of this command, VSTA, does not follow the defined Long Form to Short Form Truncation Rules (see page 566). The normal short form, VST, would be the same for both VSTArt and VSTOp, so sending VST for the VSTArt command produces an error.

NOTE

The :MEASure:VSTArt command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:Y1Position command (see page 240) instead.
:MEASure:VSTOp

(see page 564)

Command Syntax
:MEASure:VSTOp <vstop_argument>

<vstop_argument> ::= value for Y2 cursor

The :MEASure:VSTOp command moves the vertical marker 2 (Y2 cursor) to the specified value corresponding to the selected source. The source can be selected by the MARKer:X2Y2source command.

NOTE
The short form of this command, VSTO, does not follow the defined Long Form to Short Form Truncation Rules (see page 566). The normal short form, VST, would be the same for both VSTArt and VSTOp, so sending VST for the VSTOp command produces an error.

NOTE
The :MEASure:VSTOp command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :MARKer:Y2Position command (see page 241) instead.

Query Syntax
:MEASure:VSTOp?

The :MEASure:VSTOp? query returns the current value of the Y2 cursor.

Return Format
/value/<NL>

/value> ::= value of the Y2 cursor in NR3 format

See Also
- "Introduction to :MARKer Commands" on page 233
- "Introduction to :MEASure Commands" on page 248
- ":MARKer:Y1Position" on page 240
- ":MARKer:Y2Position" on page 241
- ":MARKer:YDELta" on page 242
- ":MARKer:X2Y2source" on page 238
- ":MEASure:SOURce" on page 270
- ":MEASure:TDELta" on page 513
- ":MEASure:TSTArt" on page 517
:PRInT?

(see page 564)

Query Syntax

:PRInT? [<options>]

<options> ::= [<print option>][,...,<print option>]

<print option> ::= {COLor | GRAYscale | BMP8bit | BMP}

The :PRInT? query pulls image data back over the bus for storage.

**NOTE**

The :PRINT command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :DISPlay:DATA command (see page 196) instead.

<table>
<thead>
<tr>
<th>Print Option</th>
<th>:PRInT command</th>
<th>:PRInT? query</th>
<th>Query Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLor</td>
<td>Sets palette=COLor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRAYscale</td>
<td>Sets palette=GRAYscale</td>
<td>palette=COLor</td>
<td></td>
</tr>
<tr>
<td>PRINter0,1</td>
<td>Causes the USB printer #0,1 to be selected as destination (if connected)</td>
<td>Not used</td>
<td>N/A</td>
</tr>
<tr>
<td>BMP8bit</td>
<td>Sets print format to 8-bit BMP</td>
<td>Selects 8-bit BMP formatting for query</td>
<td>N/A</td>
</tr>
<tr>
<td>BMP</td>
<td>Sets print format to BMP</td>
<td>Selects BMP formatting for query</td>
<td>N/A</td>
</tr>
<tr>
<td>FACTors</td>
<td>Selects outputting of additional settings information for :PRINT</td>
<td>Not used</td>
<td>N/A</td>
</tr>
<tr>
<td>NOFactors</td>
<td>Deselects outputting of additional settings information for :PRINT</td>
<td>Not used</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Old Print Option:**

<table>
<thead>
<tr>
<th>Is Now:</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLor</td>
</tr>
<tr>
<td>GRAYscale</td>
</tr>
<tr>
<td>PRINter0</td>
</tr>
</tbody>
</table>
### Obsolete and Discontinued Commands

#### Old Print Option: Is Now:

<table>
<thead>
<tr>
<th>Old Print Option</th>
<th>Is Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISK</td>
<td>invalid</td>
</tr>
<tr>
<td>PCL</td>
<td>invalid</td>
</tr>
</tbody>
</table>

**NOTE**
The PRINt? query is not a core command.

**See Also**
- "Introduction to Root (:) Commands" on page 102
- "Introduction to :HARDCopy Commands" on page 224
- ":HARDCopy:FORMat" on page 228
- ":HARDCopy:FACTors" on page 225
- ":HARDCopy:GRAYscale" on page 510
- ":DISPLAY:DATA" on page 196
**Invalid Syntax**

**Command Syntax**

:TIMebase:DELay <delay_value>

<delay_value> ::= time in seconds from trigger to the delay reference point on the screen.

The valid range for delay settings depends on the time/division setting for the main time base.

The :TIMebase:DELay command sets the main time base delay. This delay is the time between the trigger event and the delay reference point on the screen. The delay reference point is set with the :TIMebase:REFerence command (see page 326).

**NOTE**

The :TIMebase:DELay command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :TIMebase:POSition command (see page 323) instead.

**Query Syntax**

:TIMebase:DELay?

The :TIMebase:DELay query returns the current delay value.

**Return Format**

<delay_value><NL>

<delay_value> ::= time from trigger to display reference in seconds in NR3 format.

**Example Code**

' TIMEBASE_DELAY - Sets the time base delay. This delay is the internal time between the trigger event and the onscreen delay reference point.

' Set time base delay to 0.0.
myScope.WriteString "::TIMEBASE:DELAY 0.0"

Example program from the start: "VISA COM Example in Visual Basic" on page 614
:TRIGger:CAN:ACKNowledge

(see page 564)

Command Syntax

:TRIGger:CAN:ACKNowledge <value>

<value> ::= (0 | OFF)

This command was used with the N2758A CAN trigger module for 54620/54640 Series mixed-signal oscilloscopes. The 6000 Series oscilloscopes do not support the N2758A CAN trigger module.

Query Syntax

:TRIGger:CAN:ACKNowledge?

The :TRIGger:CAN:ACKNowledge? query returns the current CAN acknowledge setting.

Return Format

<value><NL>

<value> ::= 0

See Also

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:TRIGger" on page 352
**:TRIGger:CAN:SIGNal:DEFinition**

(see page 564)

**Command Syntax**

`:TRIGger:CAN:SIGNal:DEFinition <value>`

`<value> ::= (CANH | CANL | RX | TX | DIFFerential)`

The `:TRIGger:CAN:SIGNal:DEFinition` command sets the CAN signal type when `:TRIGger:CAN:TRIGger` is set to SOF (start of frame). These signals can be set to:

- Dominant high signal:
  - **CANH** — the actual CAN_H differential bus signal.

- 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.
  - **DIFFerential** — the CAN differential bus signal connected to an analog source channel using a differential probe.

**NOTE**

With 6000 Series oscilloscope software version 3.50 or greater, this command is available, but the only legal value is DIFF.

**Query Syntax**

`:TRIGger:CAN:SIGNal:DEFinition?`


**Return Format**

`<value><NL>`

`<value> ::= DIFF`

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:CAN:SIGNal:BAUDrate" on page 350
- ":TRIGger:CAN:SOURce" on page 351
- ":TRIGger:CAN:TRIGger" on page 352
**:TRIGger:LIN:SIGNal:DEFinition**

(see page 564)

**Command Syntax**

:TRIGger:LIN:SIGNal:DEFinition <value>

<value> ::= (LIN | RX | TX)

The :TRIGger:LIN:SIGNal:DEFinition command sets the LIN signal type. These signals can be set to:

Dominant low signals:
- LIN — the actual LIN single-end bus signal line.
- RX — the Receive signal from the LIN bus transceiver.
- TX — the Transmit signal to the LIN bus transceiver.

**NOTE**

With 6000 Series oscilloscope software version 3.50 or greater, this command is available, but the only legal value is LIN.

**Query Syntax**

:TRIGger:LIN:SIGNal:DEFinition?


**Return Format**

<value><NL>

<value> ::= LIN

**See Also**

- "Introduction to :TRIGger Commands" on page 332
- ":TRIGger:MODE" on page 338
- ":TRIGger:LIN:SIGNal:BAUDrate" on page 403
- ":TRIGger:LIN:SOURce" on page 404
:TRIGger:THReshold

(see page 564)

Command Syntax

:TRIGger:THReshold <channel group>, <threshold type> [, <value>]

<channel group> ::= {POD1 | POD2}
<threshold type> ::= {CMOS | ECL | TTL | USERdef}
<value> ::= voltage for USERdef (floating-point number) [Volt type]
/Volt type\ ::= {V | mV | uV}

The :TRIGger:THReshold command sets the threshold (trigger level) for a pod of 8 digital channels (either digital channels 0 through 7 or 8 through 15). The threshold can be set to a predefined value or to a user-defined value. For the predefined value, the voltage parameter is not required.

NOTE

This command is only available on the MSO models.


NOTE

The :TRIGger:THReshold command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :POD<n>:THReshold command (see page 290), :DIGital<n>:THReshold command (see page 192), or :TRIGger[:EDGE]:LEVel command (see page 366) for the 6000 Series oscilloscopes.

Query Syntax

:TRIGger:THReshold? <channel group>

The :TRIGger:THReshold? query returns the voltage and threshold text for analog channel 1 or 2, or POD1 or POD2.

Return Format

<threshold type>[][, <value>]<NL>
<threshold type> ::= {CMOS | ECL | TTL | USER}
CMOS ::= 2.5V
TTL ::= 1.5V
ECL ::= -1.3V
USERdef ::= range from -8.0V to +8.0V.
<value> ::= voltage for USERdef (a floating-point number in NR1.
:TRIGger:TV:TVMode

(see page 564)

Command Syntax

:TRIGger:TV:TVMode <mode>

<mode> ::= {FIELd1 | FIELd2 | AFIELds | ALINes | LINE | VERTical
            | LFIELd1 | LFIELd2 | LALTernate | LVERtical}

The :TRIGger:TV:MODE command selects the TV trigger mode and field. The
VERTical parameter is only available when :TRIGger:TV:STANdard is
GENeric. The LALTernate parameter is not available when
:TRIGger:TV:STANdard is GENeric (see page 430).

Old forms for <mode> are accepted:

<table>
<thead>
<tr>
<th>&lt;mode&gt;</th>
<th>Old Forms Accepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIELd1</td>
<td>F1</td>
</tr>
<tr>
<td>FIELd2</td>
<td>F2</td>
</tr>
<tr>
<td>AFIELd</td>
<td>ALLFields, ALLFLDS</td>
</tr>
<tr>
<td>ALINes</td>
<td>ALLLines</td>
</tr>
<tr>
<td>LFIELd1</td>
<td>LINEF1, LINEFIELD1</td>
</tr>
<tr>
<td>LFIELd2</td>
<td>LINEF2, LINEFIELD2</td>
</tr>
<tr>
<td>LALTernate</td>
<td>LINEAlt</td>
</tr>
<tr>
<td>LVERtical</td>
<td>LINEVert</td>
</tr>
</tbody>
</table>

NOTE

The :TRIGger:TV:TVMode command is an obsolete command provided for compatibility to
previous oscilloscopes. Use the :TRIGger:TV:MODE command (see page 427) instead.

Query Syntax

:TRIGger:TV:TVMode?

The :TRIGger:TV:TVMode? query returns the TV trigger mode.

Return Format

<value><NL>

<value> ::= {FIEL1 | FIEL2 | AFIEL | ALIN | LINE | VERT | LFIEL1 | LFIEL2
            | LALT | LVER}
6 Error Messages

-440, Query UNTERMINATED after indefinite response

-430, Query DEADLOCKED

-420, Query UNTERMINATED

-410, Query INTERRUPTED

-400, Query error

-340, Calibration failed

-330, Self-test failed

-321, Out of memory

-320, Storage fault

-315, Configuration memory lost
-314, Save/recall memory lost

-313, Calibration memory lost

-311, Memory error

-310, System error

-300, Device specific error

-278, Macro header not found

-277, Macro redefinition not allowed

-276, Macro recursion error

-273, Illegal macro label

-272, Macro execution error

-258, Media protected

-257, File name error

-256, File name not found
-255, Directory full

-254, Media full

-253, Corrupt media

-252, Missing media

-251, Missing mass storage

-250, Mass storage error

-241, Hardware missing

This message can occur when a feature is unavailable or unlicensed.

For example, serial bus decode commands (which require a four-channel oscilloscope) are unavailable on two-channel oscilloscopes, and some serial bus decode commands are only available on four-channel oscilloscopes when the AMS (automotive serial decode) or LSS (low-speed serial decode) options are licensed.

-240, Hardware error

-231, Data questionable

-230, Data corrupt or stale

-224, Illegal parameter value
6 Error Messages

-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

+112, Unknown File Type

+113, Directory Not Supported
6 Error Messages
IEEE 488.2 defines data structures, commands, and common bit definitions for status reporting (for example, the Status Byte Register and the Standard Event Status Register). There are also instrument-defined structures and bits (for example, the Operation Status Event Register and the Overload Event Register).

An overview of the oscilloscope's status reporting structure is shown in the following block diagram. The status reporting structure allows monitoring specified events in the oscilloscope. The ability to monitor and report these events allows determination of such things as the status of an operation, the availability and reliability of the measured data, and more.
To monitor an event, first clear the event; then, enable the event. All of the events are cleared when you initialize the instrument.

To allow a service request (SRQ) interrupt to an external controller, enable at least one bit in the Status Byte Register (by setting, or unmasking, the bit in the Service Request Enable register).

The Status Byte Register, the Standard Event Status Register group, and the Output Queue are defined as the Standard Status Data Structure Model in IEEE 488.2-1987.

The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled with the corresponding event enable register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. The *CLS command clears all event registers and all queues except the output queue. If you send *CLS immediately after a program message terminator, the output queue is also cleared.
Status Reporting Data Structures

The following figure shows how the status register bits are masked and logically OR'ed to generate service requests (SRQ) on particular events.
The status register bits are described in more detail in the following tables:

- "Status Byte Register (STB)" on page 95
- "Standard Event Status Register (ESR)" on page 82
- "Operation Status Condition Register" on page 122
- "Operation Status Event Register" on page 124
- "Overload Event Register (OVLR)" on page 128
- "Hardware Event Condition Register" on page 115
- "Hardware Event Event Register" on page 117

The status registers picture above shows how the different status reporting data structures work together. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register.

To generate a service request (SRQ) interrupt to an external controller, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the *SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.
Status Byte Register (STB)

The Status Byte Register is the summary-level register in the status reporting structure. It contains summary bits that monitor activity in the other status registers and queues. The Status Byte Register is a live register. That is, its summary bits are set and cleared by the presence and absence of a summary bit from other event registers or queues.

If the Status Byte Register is to be used with the Service Request Enable Register to set bit 6 (RQS/MSS) and to generate an SRQ, at least one of the summary bits must be enabled, then set. Also, event bits in all other status registers must be specifically enabled to generate the summary bit that sets the associated summary bit in the Status Byte Register.

The Status Byte Register can be read using either the *STB? Common Command or the GPIB 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.

```plaintext
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.

```vbscript
varQueryResult = myScope.IO.ReadSTB
MsgBox "Status Byte Register, Serial Poll: 0x" + Hex(varQueryResult)
```

**NOTE**

*Use Serial Polling to Read Status Byte Register.* Serial polling is the preferred method to read the contents of the Status Byte Register because it resets bit 6 and allows the next enabled event that occurs to generate a new SRQ interrupt.
Service Request Enable Register (SRE)

Setting the Service Request Enable Register bits enable corresponding bits in the Status Byte Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

Bits are set in the Service Request Enable Register using the *SRE command and the bits that are set are read with the *SRE? query.

**Example**

The following example sets bit 4 (MAV) and bit 5 (ESB) in the Service Request Enable Register.

```
myScope.WriteString "*SRE " + CStr(CInt("&H30"))
```

This example uses the decimal parameter value of 48, the string returned by `CStr(CInt("&H30"))`, to enable the oscilloscope to generate an SRQ interrupt under the following conditions:

- When one or more bytes in the Output Queue set bit 4 (MAV).
- When an enabled event in the Standard Event Status Register generates a summary bit that sets bit 5 (ESB).
Trigger Event Register (TER)

This register sets the TRG bit in the status byte when a trigger event occurs.

The TER event register stays set until it is cleared by reading the register or using the *CLS command. If your application needs to detect multiple triggers, the TER event register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation, you must clear the event register each time the trigger bit is set.
Output Queue

The output queue stores the oscilloscope-to-controller responses that are generated by certain instrument commands and queries. The output queue generates the Message Available summary bit when the output queue contains one or more bytes. This summary bit sets the MAV bit (bit 4) in the Status Byte Register.

When using the Agilent 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:DAT command do not set the MSG status bit in the Status Byte Register.
(Standard) Event Status Register (ESR)

The (Standard) Event Status Register (ESR) monitors the following oscilloscope status events:

- PON - Power On
- URQ - User Request
- CME - Command Error
- EXE - Execution Error
- DDE - Device Dependent Error
- QYE - Query Error
- RQC - Request Control
- OPC - Operation Complete

When one of these events occur, the event sets the corresponding bit in the register. If the bits are enabled in the Standard Event Status Enable Register, the bits set in this register generate a summary bit to set bit 5 (ESB) in the Status Byte Register.

You can read the contents of the Standard Event Status Register and clear the register by sending the *ESR? query. The value returned is the total bit weights of all of the bits that are set at the present time.

**Example**

The following example uses the *ESR query to read the contents of the Standard Event Status Register.

```cpp
myScope.WriteString "*ESR?"
varQueryResult = myScope.ReadNumber
MsgBox "Standard Event Status Register: 0x" + Hex(varQueryResult)
```

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights.
(Standard) Event Status Enable Register (ESE)

To allow any of the (Standard) Event Status Register (ESR) bits to generate a summary bit, you must first enable that bit. Enable the bit by using the *ESE (Event Status Enable) common command to set the corresponding bit in the (Standard) Event Status Enable Register (ESE).

Set bits are read with the *ESE? query.

**Example**

Suppose your application requires an interrupt whenever any type of error occurs. The error related bits in the (Standard) Event Status Register are bits 2 through 5 (hexadecimal value 0x3C). Therefore, you can enable any of these bits to generate the summary bit by sending:

```c
myScope.WriteString "*ESE " + CStr(CInt("&H3C"))
```

Whenever an error occurs, it sets one of these bits in the (Standard) Event Status Register. Because all the error related bits are enabled, a summary bit is generated to set bit 5 (ESB) in the Status Byte Register.

If bit 5 (ESB) in the Status Byte Register is enabled (via the *SRE command), an SRQ service request interrupt is sent to the controller PC.

**NOTE**

Disabled (Standard) Event Status Register bits respond but do not generate a summary bit. (Standard) Event Status Register bits that are not enabled still respond to their corresponding conditions (that is, they are set if the corresponding event occurs). However, because they are not enabled, they do not generate a summary bit to the Status Byte Register.
Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error 350, Queue overflow. Any time the queue overflows, the least recent errors remain in the queue, and the most recent error is discarded. The length of the oscilloscope's error queue is 30 (29 positions for the error messages, and 1 position for the Queue overflow message).

The error queue is read with the :SYSTem:ERRor? query. Executing this query reads and removes the oldest error from the head of the queue, which opens a position at the tail of the queue for a new error. When all the errors have been read from the queue, subsequent error queries return "0, No error".

The error queue is cleared when:
- the instrument is powered up,
- the instrument receives the *CLS common command, or
- the last item is read from the error queue.
Operation Status Event Register (:OPERegister[:EVENt])

This register hosts the RUN bit (bit 3), the WAIT TRIG bit (bit 5), and the OVLR bit (bit 11).

- The RUN bit is set whenever the instrument goes from a stop state to a single or running state.
- The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.
- The OVLR bit is set whenever a 50Ω input overload occurs.
- If any of these bits are set, the OPER bit (bit 7) of the Status Byte Register is set. The Operation Status Event Register is read and cleared with the :OPERegister[:EVENt]? query. The register output is enabled or disabled using the mask value supplied with the OPEE command.
Operation Status Condition Register (:OPERegister:CONDition)

This register hosts the RUN bit (bit 3), the WAIT TRIG bit (bit 5), the OVLR bit (bit 11), and the HWE bit (bit 12).

- The :OPERegister:CONDition? query returns the value of the Operation Status Condition Register.
- The HWE bit (bit 12) comes from the Hardware Event Registers.
- The RUN bit is set whenever the instrument is not stopped.
- The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.
- The OVLR bit is set whenever a 50Ω input overload occurs.
Arm Event Register (AER)

This register sets bit 5 (Wait Trig bit) in the Operation Status Register and the OPER bit (bit 7) in the Status Byte Register when the instrument becomes armed.

The ARM event register stays set until it is cleared by reading the register with the AER? query or using the *CLS command. If your application needs to detect multiple triggers, the ARM event register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation when the trigger bit is set, then you must clear the event register after each time it has been set.
Hardware Event Event Register (:HWEReRegister[:EVENT])

This register hosts the Bat On bit (bit 0).

- The Bat On bit is set whenever the instrument is operating on battery power.
Hardware Event Condition Register (:HWERegister:CONDition)

This register hosts the Bat On bit (bit 0) and the PLL LOCKED bit (bit 12).

- The :HWERegister:CONDition? query returns the value of the Hardware Event Condition Register.
- The PLL LOCKED bit (bit 12) is for internal use and is not intended for general use.
- The Bat On bit is set whenever the instrument is operating on battery power.
Clearing Registers and Queues

The *CLS common command clears all event registers and all queues except the output queue. If *CLS is sent immediately after a program message terminator, the output queue is also cleared.
Status Reporting Decision Chart

Do you want to do status reporting?

yes

Reset the instrument and clear the status registers:
myScope.WriteString ""RST"
myScope.WriteString ""CLS"

no (Your programs can read the status registers instead.)

Do you want to send a Service Request (SRQ) interrupt to the controller?

yes

Do you want to report events monitored by the Standard Event Status Register?

no

Activate the instrument function that you want to monitor.

Use the "ESE common command to enable the bits you want to use to generate the ESB summary bit in the Status Byte Register.

Use the "SRE common command to enable the bits you want to generate the ROS/MSS bit in the Status Byte Register and send an SRQ to the computer. If events are monitored by the Standard Event Status Register, also enable ESB with the "SRE command.

When an interrupt occurs, interrupt handler should serial poll STB with:
varR = myScope.IO.ReadSTB

To read the Status Byte Register, use the following:
myScope.WriteString ""STB?"
varR = myScope.ReadNumber
MsgBox "STB: 0x" + Hex(varR)

This displays the hexadecimal value of the Status Byte Register.

Determine which bits in the Status Byte Register are set.

no

Use the following to read the Standard Event Status Register:
myScope.WriteString ""ESR?"
varR = myScope.ReadNumber
MsgBox "ESR: 0x" + Hex(varR)

Use the following to see if an operation is complete:
myScope.WriteString ""OPC?"
varR = myScope.ReadNumber
MsgBox "OPC: 0x" + Hex(varR)

Use the following to read the contents of the status byte:
myScope.WriteString ""STB?"
varR = myScope.ReadNumber
MsgBox "STB: 0x" + Hex(varR)

END
7 Status Reporting
8
More About Oscilloscope Commands

Command Classifications  564
Valid Command/Query Strings  565
Query Return Values  582
All Oscilloscope Commands Are Sequential  583
Command Classifications

To help you use existing programs with your oscilloscope, or use current programs with the next generation of oscilloscopes, commands are classified by the following categories:

- "Core Commands" on page 564
- "Non-Core Commands" on page 564
- "Obsolete Commands" on page 564

Core Commands

Core commands are a common set of commands that provide basic oscilloscope functionality on this oscilloscope and future Agilent 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.

Non-Core Commands

Non-core commands are commands that provide specific features, but are not universal across all 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. Agilent's intent is to continue to expand command subsystems, such as the rich and evolving trigger feature set.

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:

- "Obsolete and Discontinued Commands" on page 491
- As well as: "Commands A-Z" on page 469
Valid Command/Query Strings

- "Program Message Syntax" on page 565
- "Command Tree" on page 569
- "Duplicate Mnemonics" on page 579
- "Tree Traversal Rules and Multiple Commands" on page 579

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 566), 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.
The instruction header is one or more mnemonics separated by colons (:) that represent the operation to be performed by the instrument. The "Command Tree" on page 569 illustrates how all the mnemonics can be joined together to form a complete header.

";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 567
- "Compound Command Headers" on page 567
- "Common Command Headers" on page 568

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 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 568 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.

The program instructions within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Or-Identify) asserted in the GPIB 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.

<table>
<thead>
<tr>
<th>Long Form</th>
<th>Short form</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGe</td>
<td>RANG</td>
</tr>
<tr>
<td>PATTern</td>
<td>PATT</td>
</tr>
<tr>
<td>TIMebase</td>
<td>TIM</td>
</tr>
<tr>
<td>DELay</td>
<td>DEL</td>
</tr>
<tr>
<td>TYPE</td>
<td>TYPE</td>
</tr>
</tbody>
</table>

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:

\[ *<\text{command header}>\text{<terminator>} \]

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.

\[ <\text{program mnemonic}>\text{<separator}>\text{<data}>\text{<terminator>} \]

When a program mnemonic or query has multiple program data, a comma separates sequential program data.

\[ <\text{program mnemonic}>\text{<separator}>\text{<data>},\text{<data}>\text{<terminator>} \]

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, 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 delayed.

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.

Command Tree

The command tree shows all of the commands and the relationships of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree because they do not affect the position of the parser within the tree. When a program message terminator (<NL>, linefeed-ASCII decimal 10) or a leading colon (:) is sent to the instrument, the parser is set to the root of the command tree.

:\ (root)  
  • :ACQuire (see page 138)  
  • :AALias (see page 140)  
  • :COMPLETE (see page 141)  
  • :COUNt (see page 142)  
  • :DAALias (see page 143)  
  • :MODE (see page 144)  
  • :POINts (see page 145)  
  • :RSIGnal (see page 146)  
  • :SRATe (see page 147)  
  • :TYPE (see page 148)  
  • :ACTivity (see page 103)  
  • :AER (Arm Event Register) (see page 104)  
  • :AUToscale (see page 105)  
  • :AMODE (see page 107)  
  • :CHANnels (see page 108)  
  • :BLANk (see page 109)  
  • :BUS<n> (see page 150)  
  • :BIT<m> (see page 152)  
  • :BITS (see page 153)  
  • :CLEar (see page 155)  
  • :DISPlay (see page 156)
More About Oscilloscope Commands

- :LABel (see page 157)
- :MASK (see page 158)
- :CALibrate (see page 159)
- :DATE (see page 160)
- :LABel (see page 161)
- :STARt (see page 162)
- :STATus (see page 163)
- :SWITch (see page 164)
- :TEMPerature (see page 165)
- :TIME (see page 166)
- :CDISplay (see page 110)
- :CHANnel<n> (see page 167)
- :BWLimit (see page 170)
- :COUPLing (see page 171)
- :DISPlay (see page 172)
- :IMPedance (see page 173)
- :INVert (see page 174)
- :LABel (see page 175)
- :OFFSet (see page 176)
- :PROBe (see page 177)
  - :ID (see page 178)
  - :SKEW (see page 179)
  - :STYPe (see page 180)
- :PROTection (see page 181)
- :RANGe (see page 182)
- :SCALe (see page 183)
- :UNITs (see page 184)
- :VERNier (see page 185)
- :DIGital<n> (see page 186)
- :DISPlay (see page 188)
- :LABel (see page 189)
- :POsition (see page 190)
- :SIZE (see page 191)
- :THReshold (see page 192)
- :DIGitize (see page 111)
• :DISPlay (see page 193)
  • :CLEar (see page 195)
  • :DATA (see page 196)
  • :LABel (see page 198)
  • :LABList (see page 199)
  • :PERSistence (see page 200)
  • :SOURce (see page 201)
  • :VECTors (see page 202)
• :EXTernal (see page 203)
  • :BWLimit (see page 204)
  • :IMPedance (see page 205)
  • :PROBe (see page 206)
  • :ID (see page 207)
  • :STYPe (see page 208)
  • :PROTection (see page 209)
  • :RANGe (see page 210)
  • :UNITs (see page 211)
• :FUNCTION (see page 212)
  • :CENTer (see page 214)
  • :DISPlay (see page 215)
  • :OFFSet (see page 216)
  • :OPERation (see page 217)
  • :RANGe (see page 218)
  • :REFerence (see page 219)
  • :SCALe (see page 220)
  • :SOURce (see page 221)
  • :SPAN (see page 222)
  • :WINDow (see page 223)
• :HARDcopy (see page 224)
  • :FACTors (see page 225)
  • :FFEed (see page 226)
  • :FILEname (see page 227)
  • :FORMat (see page 228)
  • :IGColors (see page 229)
  • :PALette (see page 230)
• :PDRiver (see page 231)
• :HWEnable (Hardware Event Enable Register) (see page 113)
• :HWERegister
  • :CONDition (Hardware Event Condition Register) (see page 115)
  • [:EVENt] (Hardware Event Event Register) (see page 117)
• :MARKer (see page 232)
  • :MODE (see page 234)
  • :X1Position (see page 235)
  • :X1Y1source (see page 236)
  • :X2Position (see page 237)
  • :X2Y2source (see page 238)
  • :XDELta (see page 239)
  • :Y1Position (see page 240)
  • :Y2Position (see page 241)
  • :YDELta (see page 242)
• :MEASure (see page 243)
  • :CLEar (see page 250)
  • :COUNter (see page 251)
  • :DEFine (see page 252)
  • :DELay (see page 255)
  • :DUTYcycle (see page 257)
  • :FALLtime (see page 258)
  • :FREQuency (see page 259)
  • :NWIDth (see page 260)
  • :OVERshoot (see page 261)
  • :PERiod (see page 263)
  • :PHASe (see page 264)
  • :PREShoot (see page 265)
  • :PWIDth (see page 266)
  • :RISetime (see page 267)
  • :SDEViation (see page 268)
  • :SHOW (see page 269)
  • :SOURce (see page 270)
  • :TEDGe (see page 272)
  • :TVALue (see page 274)
- :VAMplitude (see page 276)
- :VAverage (see page 277)
- :VBASE (see page 278)
- :VMAX (see page 279)
- :VMIN (see page 280)
- :VPP (see page 281)
- :VRMS (see page 282)
- :VTIME (see page 283)
- :VTOP (see page 284)
- :XMAX (see page 285)
- :XMIN (see page 286)
- :MERGe (see page 119)
- :OPEE (Operation Status Enable Register) (see page 120)
- :OPERegister
  - :CONDition (Operation Status Condition Register) (see page 122)
  - [:EVENt] (Operation Status Event Register) (see page 124)
- :OVLenable (Overload Event Enable Register) (see page 126)
- :OVLRRegister (Overload Event Register) (see page 128)
- :POD<n> (see page 287)
- :DISPlay (see page 288)
- :SIZE (see page 289)
- :THReshold (see page 290)
- :RUN (see page 131)
- :SBUS (see page 292)
  - :BUSDoctor
    - :ADDress (see page 294)
    - :BAUDrate (see page 295)
    - :CHANnel (see page 296)
    - :MODE (see page 297)
  - :CAN
    - :COUNt
      - :ERRor (see page 298)
      - :OVERload (see page 299)
      - :RESet (see page 300)
      - :TOTal (see page 301)
More About Oscilloscope Commands

- :UTILization (see page 302)
- :DISPlay (see page 303)
- :FLEXray
  - :COUNt
    - :NULL? (see page 304)
    - :RESet (see page 305)
    - :SYNC? (see page 306)
    - :TOTal? (see page 307)
- :IIC
  - :WIDTh (see page 311)
- :Lin
  - :PARity (see page 309)
  - :MODE (see page 310)
- :SPI
  - :ASIZe (see page 308)
- :SERial (see page 132)
- :SINGle (see page 133)
- :STATus (see page 134)
- :STOP (see page 135)
- :SYSTem (see page 312)
  - :DATE (see page 313)
  - :DSP (see page 314)
  - :ERRor (see page 315)
  - :LOCK (see page 316)
  - :SETup (see page 317)
  - :TIME (see page 319)
- :TER (Trigger Event Register) (see page 136)
- :TIMebase (see page 320)
  - :MODE (see page 322)
  - :POsition (see page 323)
  - :RANGe (see page 324)
  - :REFClock (see page 325)
  - :REFerence (see page 326)
  - :SCALe (see page 327)
  - :VERNier (see page 328)
More About Oscilloscope Commands

- :WINdow
  - :POSition (see page 329)
  - :RANGe (see page 330)
  - :SCALe (see page 331)
- :TRIGger (see page 332)
  - :HFReject (see page 336)
  - :HOLDoff (see page 337)
  - :MODE (see page 338)
  - :NREJect (see page 339)
  - :PATTern (see page 340)
  - :SWEep (see page 342)
- :CAN (see page 343)
  - :ACKNowledge (see page 528)
  - :PATTern
    - :DATA (see page 345)
      - :LENGTH (see page 346)
    - :ID (see page 347)
      - :MODE (see page 348)
    - :SAMPlepoint (see page 349)
  - :SIGNal
    - :BAUDrate (see page 350)
    - :DEFinition (see page 529)
  - :SOURce (see page 351)
    - :TRIGger (see page 352)
  - :DURation (see page 354)
    - :GREaterthan (see page 355)
  - :LESSthan (see page 356)
  - :PATTern (see page 357)
  - :QUALifier (see page 358)
  - :RANGe (see page 359)
  - :EBURst (see page 360)
  - :COUNt (see page 361)
  - :IDLE (see page 362)
  - :SLOPe (see page 363)
  - [:EDGE] (see page 364)
More About Oscilloscope Commands

- :COUPling (see page 365)
- :LEVel (see page 366)
- :REJect (see page 367)
- :SLOPe (see page 368)
- :SOURce (see page 369)
- :FLEXray (see page 370)
- :ERRor
  - :TYPE (see page 371)
- :FRAMe
  - :CCBase (see page 373)
  - :CCRepetition (see page 374)
  - :ID (see page 375)
  - :TYPE (see page 376)
- :TIME
  - :CBASE (see page 377)
  - :CREPetition (see page 378)
  - :SEGment (see page 379)
  - :SLOT (see page 380)
  - :TRIGger (see page 381)
- :GLITch (see page 382)
- :GREaterthan (see page 384)
- :LESSthan (see page 385)
- :LEVel (see page 386)
- :POLarity (see page 387)
- :QUALifier (see page 388)
- :RANGE (see page 389)
- :SOURce (see page 390)
- :HFReject (see page 336)
- :HOLDoff (see page 337)
- :IIC (see page 391)
- :PATTern
  - :ADDRes (see page 392)
  - :DATA (see page 393)
  - :DATa2 (see page 394)
- :SOURce
More About Oscilloscope Commands

- :CLOCk (see page 395)
- :DATA (see page 396)
- :TRIGger
  - :QUALifier (see page 397)
  - [:TYPE] (see page 398)
- :LIN (see page 400)
- :ID (see page 401)
- :SAMPlepoint (see page 402)
- :SIGNal
  - :BAUDrate (see page 403)
  - :DEFinition (see page 530)
- :SOURce (see page 404)
- :STANdard (see page 405)
- :SYNCbreak (see page 406)
- :TRIGger (see page 407)
- :MODE (see page 338)
- :NREJect (see page 339)
- :PATTern (see page 340)
- :SEQUence (see page 408)
  - :COUNT (see page 409)
  - :EDGE (see page 410)
  - :FIND (see page 411)
  - :PATTern (see page 412)
  - :RESet (see page 413)
  - :TIMer (see page 414)
  - :TRIGger (see page 415)
- :SPI (see page 416)
  - :CLOCk
    - :SLOPe (see page 417)
    - :TIMEout (see page 418)
  - :FRAMing (see page 419)
  - :PATTern
    - :DATA (see page 420)
    - :WIDTH (see page 421)
  - :SOURce
More About Oscilloscope Commands

- :CLOck (see page 422)
- :DATA (see page 423)
- :FRAMe (see page 424)
- :SWEep (see page 342)
- :TV (see page 425)
  - :LINE (see page 426)
  - :MODE (see page 427)
  - :POLarity (see page 428)
  - :SOURce (see page 429)
  - :STANdard (see page 430)
  - :TVMode (see page 532)
- :USB (see page 431)
  - :SOURce
    - :DMINus (see page 432)
    - :DPLus (see page 433)
  - :SPEed (see page 434)
  - :TRIGger (see page 435)
- :VIEW (see page 137)
- :WAVeform (see page 436)
  - :BYTeorder (see page 444)
  - :COUNt (see page 445)
  - :DATA (see page 446)
  - :FORMat (see page 448)
  - :POINts (see page 449)
    - :MODE (see page 451)
  - :PREamble (see page 453)
  - :SOURce (see page 456)
  - :TYPE (see page 460)
  - :UNSIGNED (see page 461)
- :VIEW (see page 462)
- :XINCrement (see page 463)
- :XORigin (see page 464)
- :XREFerence (see page 465)
- :YINCrement (see page 466)
- :YORigin (see page 467)
Common Commands (IEEE 488.2)

- :YREFerence (see page 468)
- *CLS (see page 79)
- *ESE (see page 80)
- *ESR (see page 82)
- *IDN (see page 84)
- *LRN (see page 85)
- *OPC (see page 86)
- *OPT (see page 87)
- *RCL (see page 88)
- *RST (see page 89)
- *SAV (see page 92)
- *SRE (see page 93)
- *STB (see page 95)
- *TRG (see page 97)
- *TST (see page 98)
- *WAI (see page 99)

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 (see page 569). 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 Agilent 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"

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.

**NOTE**

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:

\[\text{<program mnemonic><data>;<:program mnemonic><data><terminator}>\]

For example:

myScope.WriteString "':TIMEbase:REFerence CENTer;:DISPlay:VECTors ON"

**NOTE**

The leading colon before DISPlay:VECTors ON tells the parser to go back to the root of the command tree. The parser can then see the DISPlay:VECTors ON command. The space between REFerence and CENter is required; so is the space between VECTors and ON.

Multiple commands may be any combination of compound and simple commands.
More About Oscilloscope 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 Agilent VISA COM library in Visual Basic, the controller statements:

```vbnet
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.

Read Query Results Before Sending Another Command. Sending another command or query before reading the result of a query clears the output buffer (the current response) and places a Query INTERRUPTED error in the error queue.

Infinity Representation

The representation of infinity is +9.9E+37. This is also the value returned when a measurement cannot be made.
All Oscilloscope Commands Are Sequential

IEEE 488.2 makes the distinction between sequential and overlapped commands:

- *Sequential commands* finish their task before the execution of the next command starts.
- *Overlapped commands* run concurrently. Commands following an overlapped command may be started before the overlapped command is completed.

All of the oscilloscope commands are sequential.
Example programs are ASCII text files that can be cut from the help file and pasted into your favorite text editor.
SICL Example in C

/*
* Agilent SICL Example in C
* *------------------------------------------------------------------
* This program illustrates most of the commonly-used programming
* features of your Agilent oscilloscope.
* This program is to be built as a WIN32 console application.
* Edit the DEVICE_ADDRESS line to specify the address of the
* applicable device.
*/

#include <stdio.h>    /* For printf(). */
#include "sicl.h"     /* SICL routines. */

#define DEVICE_ADDRESS "usb0[2391::5970::30D3090541::0]"     /* USB */
#define WAVE_DATA_SIZE 5000
#define TIMEOUT 5000
#define SETUP_STR_SIZE 3000
#define IMG_SIZE 300000

/* Function prototypes */
void initialize(void);    /* Initialize the oscilloscope. */
void extra(void);         /* Miscellaneous commands not executed,
                           shown for reference purposes. */
void capture(void);       /* Digitize data from oscilloscope. */
void analyze(void);       /* Make some measurements. */
void get_waveform(void);  /* Download waveform data from
                           oscilloscope. */
void save_waveform(void); /* Save waveform data to a file. */
void retrieve_waveform(void); /* Load waveform data from a file. */

/* Global variables */
INST id;                  /* Device session ID. */
char buf[256] = { 0 };    /* Buffer for IDN string. */

/* Array for waveform data. */
unsigned char waveform_data[WAVE_DATA_SIZE];
double preamble[10];      /* Array for preamble. */

void main(void)
{
    /* Install a default SICL error handler that logs an error message
     * and exits. On Windows 98SE or Windows Me, view messages with
     * the SICL Message Viewer. For Windows 2000 or XP, use the Event
     * Viewer.
     */
    ionerror(I_ERROR_EXIT);

    /* Open a device session using the DEVICE_ADDRESS */
    id = iopen(DEVICE_ADDRESS);

    /* Initialize the oscilloscope. */
    initialize();
    /* Digitize data from oscilloscope. */
    capture();
    /* Make some measurements. */
    analyze();
    /* Load waveform data from a file. */
    retrieve_waveform();
}
if (id == 0)
{
    printf ("Oscilloscope iopen failed!\n");
}
else
{
    printf ("Oscilloscope session initialized!\n");
    /* Set the I/O timeout value for this session to 5 seconds. */
    itimeout(id, TIMEOUT);
    /* Clear the interface. */
    iclear(id);
    iremote(id);
}
initialize();

/* The extras function contains miscellaneous commands that do not
 * need to be executed for the proper operation of this example.
 * The commands in the extras function are shown for reference
 * purposes only.
 */
/* extra(); */ /* <-- Uncomment to execute the extra function */
capture();
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
 * ----------------------------------------------------------
 * This function initializes both the interface and the oscilloscope
 * to a known state.
 * /
void initialize (void)
{
    /* RESET - This command puts the oscilloscope in a known state.
    * Without this command, the oscilloscope settings are unknown.
    * This command is very important for program control.
    * 
    * Many of the following initialization commands are initialized
    * by this command. It is not necessary to reinitialize them
    * unless you want to change the default setting.
    */
# Programming Examples

```c
iprintf(id, "**RST
");
/* Write the *IDN? string and send an EOI indicator, then read
* the response into buf.
ippromptf(id, "*IDN?\n", "%t", buf);
printf("%s\n", buf);
*/

/* AUTOSCALE - This command evaluates all the input signals and
* sets the correct conditions to display all of the active signals.
*/
iprintf(id, ":AUTOSCALE\n");

/* CHANNEL_PROBE - Sets the probe attenuation factor for the
* selected channel. The probe attenuation factor may be from
* 0.1 to 1000.
*/
iprintf(id, ":CHAN1:PROBE 10\n");

/* CHANNEL_RANGE - Sets the full scale vertical range in volts.
* The range value is eight times the volts per division.
*/
iprintf(id, ":CHANNEL1:RANGE 8\n");

/* TIME_RANGE - Sets the full scale horizontal time in seconds.
* The range value is ten times the time per division.
*/
iprintf(id, ":TIM:RANG 2e-3\n");

/* TIME_REFERENCE - Possible values are LEFT and CENTER:
* - LEFT sets the display reference one time division from the
* left.
* - CENTER sets the display reference to the center of the screen.
*/
iprintf(id, ":TIMEBASE:REFERENCE CENTER\n");

/* TRIGGER_SOURCE - Selects the channel that actually produces the
* TV trigger. Any channel can be selected.
*/
iprintf(id, ":TRIGGER:TV:SOURC CHANNEL1\n");

/* TRIGGER_MODE - Set the trigger mode to, EDGE, GLI Tch, PATTern,
* CAN, DURation, IIC, LIN, SEQuence, SPI, TV, or USB.
*/
iprintf(id, ":TRIGGER:MODE EDGE\n");

/* TRIGGER_EDGE_SLOPE - Set the slope of the edge for the trigger
* to either POSITIVE or NEGATIVE.
*/
iprintf(id, ":TRIGGER:EDGE:SLOPE POSITIVE\n");
```
void extra (void)
{
    /* RUN_STOP (not executed in this example):
    * - RUN starts the acquisition of data for the active waveform
    *   display.
    * - STOP stops the data acquisition and turns off AUTOSTORE.
    */
    printf(id, "RUN\\n");
    printf(id, "STOP\\n");

    /* VIEW_BLANK (not executed in this example):
    * - VIEW turns on (starts displaying) an active channel or pixel
    *   memory.
    * - BLANK turns off (stops displaying) a specified channel or
    *   pixel memory.
    */
    printf(id, "BLANK CHANNEL1\\n");
    printf(id, "VIEW CHANNEL1\\n");

    /* TIME_MODE (not executed in this example) - Set the time base
    * mode to MAIN, DELAYED, XY or ROLL.
    */
    printf(id, "TIMEBASE:MODE MAIN\\n");
}

void capture (void)
{
    /* AQUIRE_TYPE - Sets the acquisition mode. There are three
    * acquisition types NORMAL, PEAK, or AVERAGE.
    */
    printf(id, "ACQUIRE:TYPE NORMAL\\n");

    /* AQUIRE_COMPLETE - Specifies the minimum completion criteria
    * for an acquisition. The parameter determines the percentage
    * of time buckets needed to be "full" before an acquisition is
    * considered to be complete.
    */
    printf(id, "ACQUIRE:COMPLETE 100\\n");

    /* DIGITIZE - Used to acquire the waveform data for transfer over
    * the interface. Sending this command causes an acquisition to
    * take place with the resulting data being placed in the buffer.
    */

    /* NOTE! The use of the DIGITIZE command is highly recommended
    * as it will ensure that sufficient data is available for
    * measurement. Keep in mind when the oscilloscope is running,
* communication with the computer interrupts data acquisition.
* Setting up the oscilloscope over the bus causes the data
* buffers to be cleared and internal hardware to be reconfigured.
* If a measurement is immediately requested there may not have
* been enough time for the data acquisition process to collect
* data and the results may not be accurate. An error value of
* 9.9E+37 may be returned over the bus in this situation.
*/
iprintf(id, ":DIGITIZE CHAN1\n");
}

/*
* analyze
*------------------------------------------------------------------
* In this example we will do the following:
* - Save the system setup to a file for restoration at a later time.
* - Save the oscilloscope display to a file which can be printed.
* - Make single channel measurements.
*/

void analyze (void)
{
    double frequency, vpp; /* Measurements. */
double vdiv, off, sdiv, delay; /* Calculated from preamble data. */
int i; /* Loop counter. */
/* Array for setup string. */
unsigned char setup_string[SETUP_STR_SIZE];
int setup_size;
FILE *fp;
unsigned char image_data[IMG_SIZE]; /* Array for image data. */
int img_size;

/* 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,
* #800002204<setup string><NL>
* where the setup string is 2204 bytes in length.
*/
setup_size = SETUP_STR_SIZE;
/* Query and read setup string. */
ipromptf(id, ":SYSTEM:SETUP?\n", ":#b\n", &setup_size, setup_string);
printf("Read setup string query (%d bytes)\n", setup_size);
/* Write setup string to file. */
fp = fopen ("c:\scope\config\setup.dat", "wb");
setup_size = fwrite(setup_string, sizeof(unsigned char), setup_size, fp);
fclose (fp);
printf("Wrote setup string (%d bytes) to file.\n", setup_size);

/* RESTORE_SYSTEM_SETUP - Uploads a previously saved setup string
* to the oscilloscope.
*/
/* Read setup string from file. */
fp = fopen ("c:\scope\config\setup.dat", "rb");
setup_size = fread(setup_string, sizeof(unsigned char),
SETUP_STR_SIZE, fp);
fclose (fp);
}
printf("Read setup string (%d bytes) from file.\n", setup_size);
/* Restore setup string. */
iprintf(id, "SYSTEM:SETUP #8%08d", setup_size);
ifwrite(id, setup_string, setup_size, 1, &setup_size);
printf("Restored setup string (%d bytes).\n", setup_size);

/* IMAGE_TRANSFER - In this example we will query for the image
* data with ":DISPLAY:DATA?" to read the data and save the data
* to the file "image.dat" which you can then send to a printer. */
itimeout(id, 30000);
printf("Transferring image to c:\scope\data\screen.bmp\n");
img_size = IMG_SIZE;
ipromptf(id, ":DISPLAY:DATA? BMP8bit, SCREEN, COLOR\n", 
   "%#b\n",
   &img_size, image_data);
printf("Read display data query (%d bytes)\n", img_size);
/* Write image data to file. */
fp = fopen ("c:\scope\data\screen.bmp", "wb");
img_size = fwrite(image_data, sizeof(unsigned char), img_size, fp);
close (fp);
printf("Wrote image data (%d bytes) to file.\n", img_size);
itimeout(id, 5000);

/* MEASURE - The commands in the MEASURE subsystem are used to
* make measurements on displayed waveforms. */
/* Set source to measure. */
iprintf(id, ":MEASURE:SOURCE CHANNEL1\n");
/* Query for frequency. */
ipromptf(id, ":MEASURE:FREQUENCY?\n", 
   "%.lf", &frequency);
printf("The frequency is: %.4f kHz\n", frequency / 1000);
/* Query for peak to peak voltage. */
ipromptf(id, ":MEASURE:VPP?\n", 
   "%.lf", &vpp);
printf("The peak to peak voltage is: %.2f V\n", vpp);

/* WAVEFORM_DATA - Get waveform data from oscilloscope. */
get_waveform();
/* Make some calculations from the preamble data. */
vdiv = 32 * preamble [7];
off = preamble [8];
/* Print them out... */
printf ("Scope Settings for Channel 1:\n");
printf ("Volts per Division = %f\n", vdiv);
printf ("Offset = %f\n", off);
printf ("Seconds per Division = %f\n", sdiv);
printf ("Delay = %f\n", delay);
/* Print the waveform voltage at selected points */
for (i = 0; i < 1000; i = i + 50)
Programming Examples

printf ("Data Point %4d = %6.2f Volts at %10f Seconds\n", i,
   ((float)waveform_data[i] - preamble[9]) * preamble[7] +
   preamble[8],
   ((float)i - preamble[6]) * preamble[4] + preamble[5]);

save_waveform(); /* Save waveform data to disk. */
retrieve_waveform(); /* Load waveform data from disk. */
}

/*
* get_waveform
* ------------------------------------------------------------------
* This function transfers the data displayed on the oscilloscope to
* the computer for storage, plotting, or further analysis.
*/

void get_waveform (void)
{
    int waveform_size;

    /* 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:PREAMBLE?"
    * query provides information concerning the vertical and horizontal
    * scaling of the waveform data.
    *
    * With the preamble information you can then use the
    * ":WAVEFORM:DATA?" query and read the data block in the
    * correct format.
    */

    /* WAVE_FORMAT - Sets the data transmission mode for waveform data
    * output. This command controls how the data is formatted when
    * sent from the oscilloscope and can be set to WORD or BYTE format.
    */
    iprintf(id, ":WAVEFORM:FORMAT BYTE\n");

    /* WAVE_POINTS - Sets the number of points to be transferred.
    * The number of time points available is returned by the
    * ":ACQUIRE:POINTS?" query. This can be set to any binary
    * fraction of the total time points available.
    */
    iprintf(id, ":WAVEFORM:POINTS 1000\n");

    /* GET_PREAMBLE - The preamble contains all of the current WAVEFORM
    * settings returned in the form <preamble block><NL> where the
    * <preamble block> is:
    *
    * FORMAT : int16 - 0 = BYTE, 1 = WORD, 2 = 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 the x-origin.
* YINCREMENT : float32 - voltage difference between data points.
* YORIGIN : float32 - value of the voltage at center screen.
* YREFERENCE : int32 - data point where y-origin occurs.
*/
printf("Reading preamble\n");
ipromptf(id, ":WAVEFORM:PREAMBLE?\n", ",%10lf\n", preamble);
/*
printf("Preamble FORMAT: %e\n", preamble[0]);
printf("Preamble TYPE: %e\n", preamble[1]);
printf("Preamble POINTS: %e\n", preamble[2]);
printf("Preamble COUNT: %e\n", preamble[3]);
printf("Preamble XINCREMENT: %e\n", preamble[4]);
printf("Preamble XORIGIN: %e\n", preamble[5]);
printf("Preamble XREFERENCE: %e\n", preamble[6]);
printf("Preamble YINCREMENT: %e\n", preamble[7]);
printf("Preamble YORIGIN: %e\n", preamble[8]);
printf("Preamble YREFERENCE: %e\n", preamble[9]);
*/

/* QUERY_WAVE_DATA - Outputs waveform records to the controller
 * over the interface that is stored in a buffer previously
 * specified with the ":WAVEFORM:SOURCE" command.
 */
iprintf(id, ":WAVEFORM:DATA?\n"); /* Query waveform data. */

/* READ_WAVE_DATA - The wave data consists of two parts: the header,
 * and the actual waveform data followed by an New Line (NL)
 * character. The query data has the following format:
 *<header><waveform data block><NL>
 *
 * Where:
 *<header> = #800002048 (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 which can be set using the ":WAVEFORM:POINTS"
 * command. You may then read that number of bytes from the
 * oscilloscope; then, read the following NL character to
 * terminate the query.
 */
waveform_size = WAVE_DATA_SIZE;
/* Read waveform data. */
iscanf(id, "%#b\n", &waveform_size, waveform_data);
if ( waveform_size == WAVE_DATA_SIZE )
{
    printf("Waveform data buffer full: ");
    printf("May not have received all points.\n");
}
else
{
    printf("Reading waveform data... size = %d\n", waveform_size);
}
9 Programming Examples

)
/*
 * save_waveform
 * -----------------------------------------------
 * This function saves the waveform data from the get_waveform
 * function to disk. The data is saved to a file called "wave.dat".
 */

void save_waveform(void)
{
    FILE *fp;

    fp = fopen("c:\scope\data\wave.dat", "wb");
    /* Write preamble. */
    fwrite(preamble, sizeof(preamble[0]), 10, fp);
    /* Write actually waveform data. */
    fwrite(waveform_data, sizeof(waveform_data[0]),
           (int)preamble[2], fp);
    fclose (fp);
}

;/*
 * retrieve_waveform
 * -----------------------------------------------
 * This function retrieves previously saved waveform data from a
 * file called "wave.dat".
 */

void retrieve_waveform(void)
{
    FILE *fp;

    fp = fopen("c:\scope\data\wave.dat", "rb");
    /* Read preamble. */
    fread (preamble, sizeof(preamble[0]), 10, fp);
    /* Read the waveform data. */
    fread (waveform_data, sizeof(waveform_data[0]),
           (int)preamble[2], fp);
    fclose (fp);
}
/*
* Agilent VISA Example in C
* ------------------------------------------------------------------
* This program illustrates most of the commonly-used programming
* features of your Agilent oscilloscope.
* This program is to be built as a WIN32 console application.
* Edit the RESOURCE line to specify the address of the
* applicable device.
*/

#include <stdio.h>    /* For printf(). */
#include <visa.h>     /* Agilent VISA routines. */

/* GPIB */
/* #define RESOURCE "GPIB0::7::INSTR" */

/* LAN */
/* #define RESOURCE "TCPIP0::a-mso6102-90541::inst0::INSTR" */

/* USB */
#define RESOURCE "USB0::2391::5970::30D3090541::0::INSTR"

#define WAVE_DATA_SIZE 5000
#define TIMEOUT 5000
#define SETUP_STR_SIZE 3000
#define IMG_SIZE 300000

/* Function prototypes */
void initialize(void); /* Initialize the oscilloscope. */
void extra(void); /* Miscellaneous commands not executed,
                   shown for reference purposes. */
void capture(void); /* Digitize data from oscilloscope. */
void analyze(void); /* Make some measurements. */
void get_waveform(void); /* Download waveform data from
                           oscilloscope. */
void save_waveform(void); /* Save waveform data to a file. */
void retrieve_waveform(void); /* Load waveform data from a file. */

/* Global variables */
ViSession defaultRM, vi;       /* Device session ID. */
char buf[256] = { 0 };        /* Buffer for IDN string. */
unsigned char waveform_data[WAVE_DATA_SIZE]; /* Array for waveform
data. */
double preamble[10];          /* Array for preamble. */

void main(void)
{
    /* Open session. */
    viOpenDefaultRM(&defaultRM);
    viOpen(defaultRM, RESOURCE, VI_NULL, VI_NULL, &vi);
    printf ('Oscilloscope session initialized!\n');
}
/* Clear the interface. */
viClear(vi);

initialize();

/* The extras function contains miscellaneous commands that do not
 * need to be executed for the proper operation of this example.
 * The commands in the extras function are shown for reference
 * purposes only.
 */
/* extra(); */ /* <-- Uncomment to execute the extra function */
capture();
analyze();

/* Close session */
viClose(vi);
viClose(defaultRM);
printf("Program execution is complete...
");
}

/*
 * initialize
 * ------------------------------------------------------------------
 * This function initializes both the interface and the oscilloscope
 * to a known state.
 */

void initialize (void)
{
/* RESET - This command puts the oscilloscope in a known state.
 * Without this command, the oscilloscope settings are unknown.
 * This command is very important for program control.
 * 
 * Many of the following initialization commands are initialized
 * by this command. It is not necessary to reinitialize them
 * unless you want to change the default setting.
 */
viPrintf(vi, "*RST\n");

/* Write the *IDN? string and send an EOI indicator, then read
 * the response into buf.
 viQueryf(vi, "*IDN?\n", "%t", buf);
 printf("%s\n", buf);
 */

/* AUTOSCALE - This command evaluates all the input signals and
 * sets the correct conditions to display all of the active signals.
 */
viPrintf(vi, ":AUTOSCALE\n");

/* CHANNEL_PROBE - Sets the probe attenuation factor for the
 * selected channel. The probe attenuation factor may be from
 * 0.1 to 1000.
 */
viPrintf(vi, ":CHAN1:PROBE 10\n");
/* CHANNEL_RANGE - Sets the full scale vertical range in volts.  
* The range value is eight times the volts per division.  
*/
viPrintf(vi, "::CHANNEL1:RANGE 8\n");

/* TIME_RANGE - Sets the full scale horizontal time in seconds.  
* The range value is ten times the time per division.  
*/
vPrintf(vi, "::TIM:RANG 2e-3\n");

/* TIME_REFERENCE - Possible values are LEFT and CENTER:  
* - LEFT sets the display reference one time division from the  
*   left.  
* - CENTER sets the display reference to the center of the screen.  
*/
vPrintf(vi, "::TIMEBASE:REFERENCE CENTER\n");

/* TRIGGER_SOURCE - Selects the channel that actually produces the  
* TV trigger. Any channel can be selected.  
*/
vPrintf(vi, "::TRIGGER:TV:SOURCE CHANNEL1\n");

/* TRIGGER_MODE - Set the trigger mode to, EDGE, GLItch, PATTern,  
* CAN, DURation, IIC, LIN, SEQuence, SPI, TV, or USB.  
*/
vPrintf(vi, "::TRIGGER:MODE EDGE\n");

/* TRIGGER_EDGE_SLOPE - Set the slope of the edge for the trigger  
* to either POSITIVE or NEGATIVE.  
*/
vPrintf(vi, "::TRIGGER:EDGE:SLOPE POSITIVE\n");

/* extra  
* -----------------------------------------------  
* The commands in this function are not executed and are shown for  
* reference purposes only. To execute these commands, call this  
* function from main.  
*/

void extra (void)
{
/* RUN_STOP (not executed in this example):  
* - RUN starts the acquisition of data for the active waveform  
*   display.  
* - STOP stops the data acquisition and turns off AUTOSTORE.  
*/
vPrintf(vi, "::RUN\n");
vPrintf(vi, "::STOP\n");

/* VIEW_BLANK (not executed in this example):  
* - VIEW turns on (starts displaying) an active channel or pixel  
*   memory.  
* - BLANK turns off (stops displaying) a specified channel or  
*   pixel memory.  
*/
*/
viPrintf(vi, "::BLANK CHANNEL1\n");
viPrintf(vi, "::VIEW CHANNEL1\n");

/* TIME_MODE (not executed in this example) - Set the time base
* mode to MAIN, DELAYED, XY or ROLL.
*/
viPrintf(vi, "::TIMEBASE:MODE MAIN\n");
}

/
* capture
* ------------------------------------------------------------------
* This function prepares the scope for data acquisition and then
* uses the DIGITIZE MACRO to capture some data.
*/

void capture (void)
{
    /* ACQUIRE_TYPE - Sets the acquisition mode. There are three
    * acquisition types NORMAL, PEAK, or AVERAGE.
    */
    viPrintf(vi, "::ACQUIRE:TYPE NORMAL\n");

    /* 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.
    */
    viPrintf(vi, "::ACQUIRE:COMPLETE 100\n");

    /* DIGITIZE - Used to acquire the waveform data for transfer over
    * the interface. Sending this command causes an acquisition to
    * take place with the resulting data being placed in the buffer.
    */
    viPrintf(vi, "::DIGITIZE CHAN1\n");
}

/*
* analyze
* ------------------------------------------------------------------
* In this example we will do the following:
* - Save the system setup to a file for restoration at a later time.
* - Save the oscilloscope display to a file which can be printed.
* - Make single channel measurements.
*/
*/

void analyze (void)
{
    double frequency, vpp; /* Measurements. */
    double vdiv, off, sdiv, delay; /* Values calculated from preamble data. */
    int i; /* Loop counter. */
    unsigned char setup_string[SETUP_STR_SIZE]; /* Array for setup string. */
    int setup_size;
    FILE *fp;
    unsigned char image_data[IMG_SIZE]; /* Array for image data. */
    int img_size;

    /* 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, #800002204<setup string><NL> where the setup string is 2204 bytes in length. */
    setup_size = SETUP_STR_SIZE;
    /* Query and read setup string. */
    viQueryf(vi, "SYSTEM:SETUP?\n", "\b\n", &setup_size, setup_string);
    printf("Read setup string query (%d bytes).\n", setup_size);
    /* Write setup string to file. */
    fp = fopen ("c:\scope\config\setup.dat", "wb");
    setup_size = fwrite(setup_string, sizeof(unsigned char), setup_size, fp);
    fclose (fp);
    printf("Wrote setup string (%d bytes) to file.\n", setup_size);

    /* RESTORE SYSTEM SETUP - Uploads a previously saved setup string to the oscilloscope. */
    /* Read setup string from file. */
    fp = fopen ("c:\scope\config\setup.dat", "rb");
    setup_size = fread (setup_string, sizeof(unsigned char), SETUP_STR_SIZE, fp);
    fclose (fp);
    printf("Read setup string (%d bytes) from file.\n", setup_size);
    /* Restore setup string. */
    viPrintf(vi, "SYSTEM:SETUP #8%08d", setup_size);
    viBufWrite(vi, setup_string, setup_size, &setup_size);
    viPrintf(vi, "\n");
    printf("Restored setup string (%d bytes).\n", setup_size);

    /* IMAGE_TRANSFER - In this example we will query for the image data with ":DISPLAY:DATA?" to read the data and save the data to the file "image.dat" which you can then send to a printer. */
    viSetAttribute(vi, VI_ATTR_TMO_VALUE, 30000);
    printf("Transferring image to c:\scope\data\screen.bmp\n");
    img_size = IMG_SIZE;
    viQueryf(vi, "DISPLAY:DATA? BMP8bit, SCREEN, COLOR\n", "\b\n", &img_size, image_data);
    printf("Read display data query (%d bytes).\n", img_size);
}
/* Write image data to file. */
fp = fopen("c:\scope\data\screen.bmp", "wb");
img_size = fwrite(image_data, sizeof(unsigned char), img_size, fp);
fclose(fp);
printf("Wrote image data (%d bytes) to file.\n", img_size);
viSetAttribute(vi, VI_ATTR_TMO_VALUE, 5000);

/* MEASURE - The commands in the MEASURE subsystem are used to
* make measurements on displayed waveforms. */

/* Set source to measure. */
viPrintf(vi, ":MEASURE:SOURCE CHANNEL1\n");

/* Query for frequency. */
viQueryf(vi, ":MEASURE:FREQUENCY?\n", "%lf", &frequency);
printf("The frequency is: %.4f kHz\n", frequency / 1000);

/* Query for peak to peak voltage. */
viQueryf(vi, ":MEASURE:VPP?\n", "%lf", &vpp);
printf("The peak to peak voltage is: %.2f V\n", vpp);

/* WAVEFORM_DATA - Get waveform data from oscilloscope. */
get_waveform();

/* Make some calculations from the preamble data. */
vddiv = 32 * preamble[7];
off = preamble[8];

/* Print them out... */
printf("Scope Settings for Channel 1:\n");
printf("Volts per Division = %f\n", vdiv);
printf("Offset = %f\n", off);
printf("Seconds per Division = %f\n", sdiv);
printf("Delay = %f\n", delay);

/* print out the waveform voltage at selected points */
for (i = 0; i < 1000; i = i + 50)
    printf("Data Point %d = %6.2f Volts at %10f Seconds\n", i,
            (float)waveform_data[i] - preamble[9]) * preamble[7] +
            preamble[8],
            ((float)i - preamble[6]) * preamble[4] + preamble[5]);

save_waveform(); /* Save waveform data to disk. */
retrievecwaveform(); /* Load waveform data from disk. */
)

/*
* get_waveform
* --------------------------------------------------------------
* This function transfers the data displayed on the oscilloscope to
* the computer for storage, plotting, or further analysis.
*/
void get_waveform (void)
{
    int waveform_size;

    /* 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:PREAMBLE?"
    * query provides information concerning the vertical and horizontal
    * scaling of the waveform data.
    * With the preamble information you can then use the
    * ":WAVEFORM:DATA?" query and read the data block in the
    * correct format.
    */

    /* WAVE_FORMAT - Sets the data transmission mode for waveform data
    * output. This command controls how the data is formatted when
    * sent from the oscilloscope and can be set to WORD or BYTE format.
    */
    viPrintf(vi, "::WAVEFORM:FORMAT BYTE\n");

    /* WAVE_POINTS - Sets the number of points to be transferred.
    * The number of time points available is returned by the
    * "ACQUIRE:POINTS?" query. This can be set to any binary
    * fraction of the total time points available.
    */
    viPrintf(vi, "::WAVEFORM:POINTS 1000\n");

    /* GET_PREAMBLE - The preamble contains all of the current WAVEFORM
    * settings returned in the form <preamble block><NL> where the
    * <preamble block> is:
    * FORMAT : int16 - 0 = BYTE, 1 = WORD, 2 = 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 the x-origin.
    * YINCREMENT : float32 - voltage difference between data points.
    * YORIGIN : float32 - value of the voltage at center screen.
    * YREFERENCE : int32 - data point where y-origin occurs.
    */
    printf("Reading preamble\n");
    viQueryf(vi, "::WAVEFORM:PREAMBLE?\n", "%10lf\n", preamble);

    /*
    printf("Preamble FORMAT: %e\n", preamble[0]);
    printf("Preamble TYPE: %e\n", preamble[1]);
    printf("Preamble POINTS: %e\n", preamble[2]);
    printf("Preamble COUNT: %e\n", preamble[3]);
    printf("Preamble XINCREMENT: %e\n", preamble[4]);
    printf("Preamble XORIGIN: %e\n", preamble[5]);
    printf("Preamble XREFERENCE: %e\n", preamble[6]);"
printf("Preamble YINCREMENT: %e\n", preamble[7]);
printf("Preamble YORIGIN: %e\n", preamble[8]);
printf("Preamble YREFERENCE: %e\n", preamble[9]);
*/

/* QUERY_WAVE_DATA - Outputs waveform records to the controller
 * over the interface that is stored in a buffer previously
 * specified with the ":WAVEFORM:SOURCE" command.
 */
viPrintf(vi, ":WAVEFORM:DATA?\n"); /* Query waveform data. */

/* READ_WAVE_DATA - The wave data consists of two parts: the header,
 * and the actual waveform data followed by an New Line (NL)
 * character. The query data has the following format:
 *
 *  <header><waveform data block><NL>
 *
 * Where:
 *
 *  <header> = #800002048   (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 which can be set using the ":WAVEFORM:POINTS"
 * command. You may then read that number of bytes from the
 * oscilloscope; then, read the following NL character to
 * terminate the query.
 */
wordform_size = WAVE_DATA_SIZE;
/* Read waveform data. */
viScanf(vi, "%#b\n", &wordform_size, waveform_data);
if ( wordform_size == WAVE_DATA_SIZE )
{
    printf("Waveform data buffer full: ");
    printf("May not have received all points.\n");
}
else
{
    printf("Reading waveform data... size = %d\n", waveform_size);
}
}

/*
 * save_waveform
 * -----------------------------------------------
 * This function saves the waveform data from the get_waveform
 * function to disk. The data is saved to a file called "wave.dat".
 */
void save_waveform(void)
{
    FILE *fp;

    fp = fopen("c:\scope\data\wave.dat", "wb"); /* Write preamble. */
    fwrite(preamble, sizeof(preamble[0]), 10, fp);
/* Write actually waveform data. */
fwrite(waveform_data, sizeof(waveform_data[0]), (int)preamble[2],
     fp);
fclose(fp);
}

/*
* retrieve_waveform
* ---------------------------------------------------------------
* This function retrieves previously saved waveform data from a
* file called "wave.dat".
*/

void retrieve_waveform(void)
{
    FILE *fp;

    fp = fopen("c:\scope\data\wave.dat", "rb");
    /* Read preamble. */
    fread(preamble, sizeof(preamble[0]), 10, fp);
    /* Read the waveform data. */
    fread(waveform_data, sizeof(waveform_data[0]), (int)preamble[2],
          fp);
    fclose(fp);
}
VISA Example in Visual Basic

Option Explicit

Public err As Long ' Error returned by VISA function calls.
Public drm As Long ' Session to Default Resource Manager.
Public vi As Long ' Session to instrument.

Public dblQueryResult As Double
Public Const DblArraySize = 20
Public Const ByteArraySize = 500000
Public retCount As Long
Public dblArray(DblArraySize) As Double
Public byteArray(ByteArraySize) As Byte
Public paramsArray(2) As Long

Public strQueryResult As String * 200

Sub Main()

' This example shows the fundamental parts of a program (initialize,
capture, analyze).

' The commands sent to the oscilloscope are written in both long and
short form. Both forms are acceptable.

' The input signal is the probe compensation signal from the front
panel of the oscilloscope connected to channel 1.

' If you are using a different signal or different channels, these
commands may not work as explained in the comments.

Sub Main()
' VISA Interface.
' "GPIB0::7::INSTR" is the address string for the device —
' this address will be the same as seen in:
' Start->Programs->Agilent IO Libraries->VISA Assistant
' (after the VISA Interface Name is defined in IO Config).

' err = viOpen(drm, "GPIB0::7::INSTR", 0, 0, vi)
' err = viOpen(drm, "TCPIP0::a-mso6102-90541::inst0::INSTR", 0, 0, vi)
err = viOpen(drm, _
    "USB0::2391::5970::30D3090541::0::INSTR", 0, 60000, vi)

' Initialize — Initialization will start the program with the
' oscilloscope in a known state.
Initialize

' Capture — After initialization, you must make waveform data
' available to analyze. To do this, capture the data using the
' DIGITIZE command.
Capture

' Analyze — Once the waveform has been captured, it can be analyzed.
' There are many parts of a waveform to analyze. This example shows
' some of the possible ways to analyze various parts of a waveform.
Analyze

' Close the vi session and the resource manager session.
err = viClose(vi)
err = viClose(drm)

End Sub

',
' Initialize
',-----------------------------------------------
' Initialize will start the program with the oscilloscope in a known
' state. This is required because some uninitialized conditions could
' cause the program to fail or not perform as expected.
',
' In this example, we initialize the following:
' - Oscilloscope
' - Channel 1 range
' - Display Grid
' - Timebase reference, range, and delay
' - Trigger mode and type
',
' There are also some additional initialization commands, which are
' not used, but shown for reference.
',-----------------------------------------------

Private Sub Initialize()

' Clear the interface.
err = viClear(vi)

' 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.

Reset the oscilloscope to the defaults.
```
err = viVPrintf(vi, "*RST" + vbLf, 0)
```

*IDN - Ask for the device's *IDN string.
```
err = viVPrintf(vi, "*IDN?" + vbLf, 0)
err = viVScanf(vi, "%t", strQueryResult) ' Read the results as a string.
```

Display results.
```
MsgBox "Result is: " + strQueryResult, vbOKOnly, "*IDN? Result"
```

AUTOSCALE - This command evaluates all the input signals and sets the correct conditions to display all of the active signals.
```
err = viVPrintf(vi, ":AUTOSCALE" + vbLf, 0) ' Same as pressing the Autoscale key.
```

CHANNEL_PROBE - Sets the probe attenuation factor for the selected channel. The probe attenuation factor may be set from 0.1 to 1000.
```
Set Probe to 10:1.
err = viVPrintf(vi, ":CHAN1:PROBE 10" + vbLf, 0)
```

CHANNEL_RANGE - Sets the full scale vertical range in volts. The range value is 8 times the volts per division.
```
Set the vertical range to 8 volts.
err = viVPrintf(vi, ":CHANNEL1:RANGE 8" + vbLf, 0)
```

TIME_RANGE - Sets the full scale horizontal time in seconds. The range value is 10 times the time per division.
```
Set the time range to 0.002 seconds.
err = viVPrintf(vi, ":TIM:RANG 2e-3" + vbLf, 0)
```

TIME_REFERENCE - Possible values are LEFT and CENTER. - LEFT sets the display reference on time division from the left. - CENTER sets the display reference to the center of the screen.
```
Set reference to center.
err = viVPrintf(vi, ":TIMEBASE:REFERENCE CENTER" + vbLf, 0)
```

TRIGGER_TV_SOURCE - Selects the channel that actually produces the TV trigger. Any channel can be selected.
```
err = viVPrintf(vi, ":TRIGGER:TV:SOURCE CHANNEL1" + vbLf, 0)
```

TRIGGER_MODE - Set the trigger mode to EDGE, GLITch, PATTern, CAN, DURation, IIC, LIN, SEQuence, SPI, TV, or USB.
```
Set the trigger mode to EDGE.
err = viVPrintf(vi, ":TRIGGER:MODE EDGE" + vbLf, 0)
```

TRIGGER_Edge_SLOPE - Sets the slope of the edge for the trigger.
```
Set the slope to positive.
err = viVPrintf(vi, ":TRIGGER:EDGE:SLOPE POSITIVE" + vbLf, 0)
```
' The following commands are not executed and are shown for reference
' purposes only. To execute these commands, uncomment them.

' RUN_STOP - (not executed in this example)
' - RUN starts the acquisition of data for the active waveform display.
' - STOP stops the data acquisition and turns off AUTOSTORE.

' Start data acquisition.
' err = viVPrintf(vi, "RUN" + vbCrLf, 0)

' Stop the data acquisition.
' err = viVPrintf(vi, "STOP" + vbCrLf, 0)

' VIEW_BLANK - (not executed in this example)
' - VIEW turns on (starts displaying) a channel or pixel memory.
' - BLANK turns off (stops displaying) a channel or pixel memory.

' Turn channel 1 off.
' err = viVPrintf(vi, "BLANK CHANNEL1" + vbCrLf, 0)

' Turn channel 1 on.
' err = viVPrintf(vi, "VIEW CHANNEL1" + vbCrLf, 0)

' TIMEBASE_MODE - (not executed in this example)
' Set the time base mode to MAIN, DELAYED, XY, or ROLL.

' Set time base mode to main.
' err = viVPrintf(vi, "TIMEBASE:MODE MAIN" + vbCrLf, 0)

End Sub

' Capture
' -------------------------------------------------------------------
' We will capture the waveform using the digitize command.
' -------------------------------------------------------------------

Private Sub Capture()

' ACQUIRE_TYPE - Sets the acquisition mode, which can be NORMAL,
' PEAK, or AVERAGE.
err = viVPrintf(vi, "ACQUIRE:TYPE NORMAL" + vbCrLf, 0)

' 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.
err = viVPrintf(vi, "ACQUIRE:COMPLETE 100" + vbCrLf, 0)

' DIGITIZE - Used to acquire the waveform data for transfer over
' the interface. Sending this command causes an acquisition to
' take place with the resulting data being placed in the buffer.

' NOTE! The DIGITIZE command is highly recommended for triggering
' modes other than SINGLE. This ensures that sufficient data is
' available for measurement. If DIGITIZE is used with single mode,
the completion criteria may never be met. The number of points
gathered in Single mode is related to the sweep speed, memory
depth, and maximum sample rate. For example, take an oscilloscope
with a 1000-point memory, a sweep speed of 10 us/div (100 us
total time across the screen), and a 20 MSa/s maximum sample rate.
1000 divided by 100 us equals 10 MSa/s. Because this number is
less than or equal to the maximum sample rate, the full 1000 points
will be digitized in a single acquisition. Now, use 1 us/div
(10 us across the screen). 1000 divided by 100 us equals 100 MSa/s;
because this is greater than the maximum sample rate by 5 times,
only 400 points (or 1/5 the points) can be gathered on a single
trigger. Keep in mind when the oscilloscope is running,
communication with the computer interrupts data acquisition.
Setting up the oscilloscope over the bus causes the data buffers
to be cleared and internal hardware to be reconfigured. If a
measurement is immediately requested, there may have not been
enough time for the data acquisition process to collect data,
and the results may not be accurate. An error value of 9.9E+37
may be returned over the bus in this situation.

err = viVPrintf(vi, ":DIGITIZE CHAN1" + vbLf, 0)

End Sub

' Analyze

' In analyze, we will do the following:
' - Save the system setup to a file and restore it.
' - Save the waveform data to a file on the computer.
' - Make single channel measurements.
' - Save the oscilloscope display to a file that can be sent to a
  printer.

Private Sub Analyze()

' Set up arrays for multiple parameter query returning an array
' with viVScanf/viVQueryf. Set retCount to the maximum number
' of elements that the array can hold.
paramsArray(0) = VarPtr(retCount)
paramsArray(1) = VarPtr(byteArray(0))

' 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,
' #800002204<setup string><NL>
'where the setup string is 2204 bytes in length.
Dim lngSetupStringSize As Long
err = viVPrintf(vi, ":SYSTEM:SETUP?" + vbLf, 0)
retCount = ByteArraySize
lngSetupStringSize = retCount

' Unsigned integer bytes.
err = viVScanf(vi, ":\%b\"n" + vbLf, paramsArray(0))
' Output setup string to a file:
Dim strPath As String
Dim lngI As Long
strPath = "c:\scope\config\setup.dat"
Close #1 ' If #1 is open, close it.

' Open file for output.
Open strPath For Binary Access Write Lock Write As #1
For lngI = 0 To lngSetupStringSize - 1
    Put #1, , byteArray(lngI) ' Write data.
Next lngI
Close #1 ' Close file.

' 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.
err = viVPrintf(vi, ":DISPLAY:DATA? BMP, SCREEN, COLOR" + vbLf, 0)
retCount = ByteArraySize

' Unsigned integer bytes.
err = viVScanf(vi, " %#b" + vbLf, paramsArray(0))

' Output display data to a file:
strPath = "c:\scope\data\screen.bmp"
' Remove file if it exists.
If Len(Dir(strPath)) Then
    Kill strPath
End If
Close #1 ' If #1 is open, close it.

' Open file for output.
Open strPath For Binary Access Write Lock Write As #1
For lngI = 0 To retCount - 1
    Put #1, , byteArray(lngI) ' Write data.
Next lngI
Close #1 ' Close file.

' RESTORE_SYSTEM_SETUP - Read the setup string from a file and write
' it back to the oscilloscope.
strPath = "c:\scope\config\setup.dat"
Open strPath For Binary Access Read As #1 ' Open file for input.
Get #1, , byteArray ' Read data.
Close #1 ' Close file.
' Write learn string back to oscilloscope using ":SYSTEM:SETUP" command:
retCount = lngSetupStringSize
err = viVPrintf(vi, ":SYSTEM:SETUP %#b" + vbLf, paramsArray(0))

' MEASURE - The commands in the MEASURE subsystem are used to make
' measurements on displayed waveforms.

' Source to measure
err = viVPrintf(vi, " :MEASURE:SOURCE CHANNEL1" + vbLf, 0)

' Query for frequency.
err = viVPrintf(vi, " :MEASURE: FREQUENCY?" + vbLf, 0)
' Read frequency.
err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblQueryResult))
MsgBox "Frequency:" + vbCrLf + _
FormatNumber(dblQueryResult / 1000, 4) + " kHz"

' Query for duty cycle.
err = viVPrintf(vi, "MEASURE:DUTYCYCLE?" + vbLf, 0)
' Read duty cycle.
err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblQueryResult))
MsgBox "Duty cycle:" + vbCrLf + FormatNumber(dblQueryResult, 3) + "%"

' Query for risetime.
err = viVPrintf(vi, "MEASURE:RISETIME?" + vbLf, 0)
' Read risetime.
err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblQueryResult))
MsgBox "Risetime:" + vbCrLf + FormatNumber(dblQueryResult * 1000000, 4) + " us"

' Query for Peak to Peak voltage.
err = viVPrintf(vi, "MEASURE:VPP?" + vbLf, 0)
' Read VPP.
err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblQueryResult))
MsgBox "Peak to peak voltage:" + vbCrLf + FormatNumber(dblQueryResult, 4) + " V"

' Query for Vmax.
err = viVPrintf(vi, "MEASURE:VMAX?" + vbLf, 0)
' Read Vmax.
err = viVScanf(vi, "%lf" + vbLf, VarPtr(dblQueryResult))
MsgBox "Maximum voltage:" + vbCrLf + FormatNumber(dblQueryResult, 4) + " V"

' 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.
err = viVPrintf(vi, "*:WAVEFORM:SOURCE CHAN1" + vbLf, 0)

' WAVE_POINTS - Specifies the number of points to be transferred
' using the "*:WAVEFORM:DATA?" query.
err = viVPrintf(vi, "*:WAVEFORM:POINTS 1000" + vbLf, 0)

' 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.
err = viVPrintf(vi, "*:WAVEFORM:FORMAT WORD" + vbLf, 0)
lngVSteps = 65536
intBytesPerData = 2

' Data in range 0 to 255.
'err = viVPrintf(vi, "*:WAVEFORM:FORMAT BYTE" + vbLf, 0)
'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, 2 = 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 difference between data points.
  YORIGIN : float32 - value is the voltage at center screen.
  YREFERENCE : int32 - specifies the data point where y-origin
    occurs.

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

' Query for the preamble.
err = viVPrintf(vi, ":WAVEFORM:PREAMBLE?" + vbLf, 0)
paramsArray(1) = VarPtr(dblArray(0))
retCount = DblArraySize

' Read preamble information.
err = viVScanf(vi, "%,#lf" + vbLf, paramsArray(0))
intFormat = dblArray(0)
intType = dblArray(1)
lngPoints = dblArray(2)
lngCount = dblArray(3)
dblXIncrement = dblArray(4)
dblXOrigin = dblArray(5)
lngXReference = dblArray(6)
sngYIncrement = dblArray(7)
sngYOrigin = dblArray(8)
lngYReference = dblArray(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 = " + _
Programming Examples

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.
' Query the oscilloscope for the waveform data.
err = viVPrintf(vi, ":WAV:DATA?" + vbCrLf, 0)
'
' 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
paramsArray(1) = VarPtr(byteArray(0))
retCount = ByteArraySize
' Unsigned integer bytes.
err = viVScanf(vi, "%#b" + vbCrLf, paramsArray(0))
' retCount is now actual number of bytes returned by query.
For lngI = 0 To retCount - 1 Step (retCount / 20) ' 20 points.
    If intBytesPerData = 2 Then
        lngDataValue = CLng(byteArray(lngI)) * 256 + _
            CLng(byteArray(lngI + 1)) ' 16-bit value.
    Else
        lngDataValue = CLng(byteArray(lngI)) ' 8-bit value.
    End If
    strOutput = strOutput + "Data point " + _
        FormatNumber(lngDataValue) + vbCrLf
End For
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(sngYOrigin) + " V" + vbCrLf
strOutput = strOutput + "Sec/Div = " + _
    FormatNumber(lngPoints * dblXIncrement / 10 * _
    1000000) + " us" + vbCrLf
strOutput = strOutput + "Delay = " + _
    FormatNumber(((lngPoints / 2) * _
    dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
CStr(lngI / intBytesPerData) + ", " + FormatNumber((lngDataValue - lngYReference) * sngYIncrement + sngYOrigin) + " V, " + FormatNumber(((lngI / intBytesPerData - lngXReference) * dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

' 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.
err = viVPrintf(vi, ":MEASURE:TEDGE? 1, CHAN1" + vbCrLf, 0)

' Read time at edge 1 on ch 1.
err = viVScanf(vi, "%lf", VarPtr(dblChan1Edge1))

' Query time at 1st rising edge on ch2.
err = viVPrintf(vi, ":MEASURE:TEDGE? 1, CHAN2" + vbCrLf, 0)

' Read time at edge 1 on ch 2.
err = viVScanf(vi, "%lf", VarPtr(dblChan2Edge1))

' 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.
err = viVPrintf(vi, ":MEASURE:TEDGE? 2, CHAN1" + vbCrLf, 0)

' Read time at edge 2 on ch 1.
err = viVScanf(vi, "%lf", VarPtr(dblChan1Edge2))

' Calculate period of ch 1.
dblPeriod = dblChan1Edge2 - dblChan1Edge1

' Calculate phase difference between ch1 and ch2.
dblPhase = (dblDelay / dblPeriod) * 360
MsgBox "Phase = " + vbCrLf + CStr(dblPhase)

End Sub
VISA COM Example in Visual Basic

```
' Agilent VISA COM Example in Visual Basic
' -------------------------------------------------------------------
' This program illustrates most of the commonly used programming
' features of your Agilent oscilloscopes.
' -------------------------------------------------------------------
Option Explicit

Public myMgr As VisaComLib.ResourceManager
Public myScope As VisaComLib.FormattedIO488
Public varQueryResult As Variant
Public strQueryResult As String

' MAIN PROGRAM
' -------------------------------------------------------------------
' This example shows the fundamental parts of a program (initialize,
' capture, analyze).

' The commands sent to the oscilloscope are written in both long and
' short form. Both forms are acceptable.

' The input signal is the probe compensation signal from the front
' panel of the oscilloscope connected to channel 1.

' If you are using a different signal or different channels, these
' commands may not work as explained in the comments.
' -------------------------------------------------------------------

Sub Main()

On Error GoTo VisaComError

' Create the VISA COM I/O resource.
Set myMgr = New VisaComLib.ResourceManager
Set myScope = New VisaComLib.FormattedIO488

' GPIB.
'Set myScope.IO = myMgr.Open("GPIB0::7::INSTR")

' LAN.
'Set myScope.IO = myMgr.Open("TCPIP0::a-mso6102-90541::inst0::INSTR")

' USB.
Set myScope.IO = myMgr.Open("USB0::2391::5970::30D3090541::0::INSTR")

' Initialize - Initialization will start the program with the
' oscilloscope in a known state.
Initialize

' Capture - After initialization, you must make waveform data
' available to analyze. To do this, capture the data using the
```
' DIGITIZE command.
Capture

' Analyze - Once the waveform has been captured, it can be analyzed.
' There are many parts of a waveform to analyze. This example shows
' some of the possible ways to analyze various parts of a waveform.
Analyze

Exit Sub

VisaComError:
MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

' Initialize
' -------------------------------------------------------------------
' Initialize will start the program with the oscilloscope in a known
' state. This is required because some uninitialized conditions could
' cause the program to fail or not perform as expected.

' In this example, we initialize the following:
' - Oscilloscope
' - Channel 1 range
' - Display Grid
' - Timebase reference, range, and delay
' - Trigger mode and type
'
' There are also some additional initialization commands, which are
' not used, but shown for reference.
' -------------------------------------------------------------------

Private Sub Initialize()

On Error GoTo VisaComError

' Clear the interface.
myScope.IO.Clear

' 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.

' AUTOSCALE - This command evaluates all the input signals and sets
' the correct conditions to display all of the active signals.

' Same as pressing the Autoscale key.
myScope.WriteString ":AUTOSCALE"

' CHANNEL_PROBE - Sets the probe attenuation factor for the selected
' channel. The probe attenuation factor may be set from 0.1 to 1000.
myScope.WriteString ":CHAN1:PROBE 10"    ' Set Probe to 10:1.
' 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"

' 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"

' TIME_REFERENCE - Possible values are LEFT and CENTER.
  - LEFT sets the display reference on time division from the left.
  - CENTER sets the display reference to the center of the screen.
myScope.WriteString " :TIMEBASE:REFERENCE CENTER"

' TRIGGER_TV_SOURCE - Selects the channel that actually produces the TV trigger. Any channel can be selected.
myScope.WriteString " :TRIGGER:TV:SOURCE CHANNEL1"

' TRIGGER_MODE - Set the trigger mode to EDGE, GLITCH, PATTERN, CAN, DURATION, IIC, LIN, SEQUENCE, SPI, TV, or USB.
myScope.WriteString " :TRIGGER:MODE EDGE"

' TRIGGER_EDGE_SLOPE - Sets the slope of the edge for the trigger.
myScope.WriteString " :TRIGGER:EDGE:SLOPE POSITIVE"

' The following commands are not executed and are shown for reference purposes only. To execute these commands, uncomment them.

' RUN_STOP - (not executed in this example)
  - RUN starts the acquisition of data 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.

' VIEW_BLANK - (not executed in this example)
  - VIEW turns on (starts displaying) a channel or pixel memory.
  - BLANK turns off (stops displaying) a channel or pixel memory.
myScope.WriteString " :BLANK CHANNEL1" ' Turn channel 1 off.
myScope.WriteString " :VIEW CHANNEL1" ' Turn channel 1 on.

' TIMEBASE_MODE - (not executed in this example)
  - Set the time base mode to MAIN, DELAYED, XY, or ROLL.
myScope.WriteString " :TIMEBASE:MODE MAIN"

Exit Sub
VisaComError:
   MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

Capture
   ' We will capture the waveform using the digitize command.
   ' -------------------------------------------------------------------------
   ' Capture
   ' -------------------------------------------------------------------
   Private Sub Capture()

   On Error GoTo VisaComError

   ' AQUIRE_TYPE - Sets the acquisition mode, which can be NORMAL, PEAK, or AVERAGE.
   myScope.WriteString ":ACQUIRE:TYPE NORMAL"

   ' AQUIRE_COMPLETE - Specifies the minimum completion criteria for an acquisition. The parameter determines the percentage of time buckets needed to be "full" before an acquisition is considered to be complete.
   myScope.WriteString ":ACQUIRE:COMPLETE 100"

   ' DIGITIZE - Used to acquire the waveform data for transfer over the interface. Sending this command causes an acquisition to take place with the resulting data being placed in the buffer.
   ' NOTE! The DIGITIZE command is highly recommended for triggering modes other than SINGLE. This ensures that sufficient data is available for measurement. If DIGITIZE is used with single mode, the completion criteria may never be met. The number of points gathered in Single mode is related to the sweep speed, memory depth, and maximum sample rate. For example, take an oscilloscope with a 1000-point memory, a sweep speed of 10 us/div (100 us total time across the screen), and a 20 MSa/s maximum sample rate. 1000 divided by 100 us equals 10 MSa/s. Because this number is less than or equal to the maximum sample rate, the full 1000 points will be digitized in a single acquisition. Now, use 1 us/div (10 us across the screen). 1000 divided by 10 us equals 100 MSa/s; because this is greater than the maximum sample rate by 5 times, only 400 points (or 1/5 the points) can be gathered on a single trigger. Keep in mind when the oscilloscope is running, communication with the computer interrupts data acquisition. Setting up the oscilloscope over the bus causes the data buffers to be cleared and internal hardware to be reconfigured. If a measurement is immediately requested, there may have not been enough time for the data acquisition process to collect data, and the results may not be accurate. An error value of 9.9E+37 may be returned over the bus in this situation.

   myScope.WriteString ":DIGITIZE CHAN1"

   Exit Sub
VisaComError:
    MsgBox "VISA COM Error:" + vbCrLf + Err.Description

End Sub

Private Sub Analyze()

    ' 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,
    ' #800002204<setup string><NL>
    ' where the setup string is 2204 bytes in length.
    myScope.WriteString " :SYSTEM:SETUP?"
    varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
    CheckForInstrumentErrors ' After reading query results.
    ' Output setup string to a file:
    Dim strPath As String
    strPath = "c:\scope\config\setup.dat"
    Close #1 ' If #1 is open, close it.
    ' Open file for output.
    Open strPath For Binary Access Write Lock Write As #1
    Put #1, , varQueryResult ' Write data.
    Close #1 ' Close file.

    ' IMAGE_TRANSFER - In this example, we will query for the image data with ":DISPLAY:DATA?", read the data, and then save it to a file.
    Dim byteData() As Byte
    myScope.IO.Timeout = 15000
    myScope.WriteString ":DISPLAY:DATA? BMP, SCREEN, COLOR"
    byteData = myScope.ReadIEEEBlock(BinaryType_UI1)
    ' Output display data to a file:
    strPath = "c:\scope\data\screen.bmp"
    ' Remove file if it exists.
    If Len(Dir(strPath)) Then
        Kill strPath
    End If
    Close #1 ' If #1 is open, close it.
    ' Open file for output.
    Open strPath For Binary Access Write Lock Write As #1
    Put #1, , byteData ' Write data.
    Close #1 ' Close file.
    myScope.IO.Timeout = 5000

    ' RESTORE_SYSTEM_SETUP - Read the setup string from a file and write
' it back to the oscilloscope.
Dim varSetupString As Variant
strPath = "c:\scope\config\setup.dat"
Open strPath For Binary Access Read As #1 ' Open file for input.
Get #1, , varSetupString ' Read data.
Close #1 ' Close file.
' Write setup string back to oscilloscope using ":SYSTEM:SETUP"
' command:
myScope.WriteIEEEBlock ":SYSTEM:SETUP ", varSetupString
CheckForInstrumentErrors

' MEASURE - The commands in the MEASURE subsystem are used to make
' measurements on displayed waveforms.

' Source to measure.
myScope.WriteString ":MEASURE:SOURCE CHANNEL1"

' Query for frequency.
myScope.WriteString ":MEASURE:FREQUENCY?"
varQueryResult = myScope.ReadNumber ' Read frequency.
MsgBox "Frequency:" + vbCrLf + 
    FormatNumber(varQueryResult / 1000, 4) + " kHz"

' Query for duty cycle.
myScope.WriteString ":MEASURE:DUTYCYCLE?"
varQueryResult = myScope.ReadNumber ' Read duty cycle.
MsgBox "Duty cycle:" + vbCrLf + 
    FormatNumber(varQueryResult, 3) + "%"

' Query for risetime.
myScope.WriteString ":MEASURE:RISETIME?"
varQueryResult = myScope.ReadNumber ' Read risetime.
MsgBox "Risetime:" + vbCrLf + 
    FormatNumber(varQueryResult * 1000000, 4) + " us"

' Query for Peak to Peak voltage.
myScope.WriteString ":MEASURE:VPP?"
varQueryResult = myScope.ReadNumber ' Read VPP.
MsgBox "Peak to peak voltage:" + vbCrLf + 
    FormatNumber(varQueryResult, 4) + " V"

' Query for Vmax.
myScope.WriteString ":MEASURE:VMAX?"
varQueryResult = myScope.ReadNumber ' Read Vmax.
MsgBox "Maximum voltage:" + vbCrLf + 
    FormatNumber(varQueryResult, 4) + " V"

' 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
' 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, 2 = 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 difference 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(sngYOrigin) + " V" + vbCrLf
' strOutput = strOutput + "Sec/Div = " + _
' FormatNumber(lngPoints * dblXIncrement / 10 * _
' 1000000) + " us" + vbCrLf
' strOutput = strOutput + "Delay = " + _
' FormatNumber(((lngPoints / 2) * _
' dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString "::WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
' <header><waveform_data><NL>
' Where:
' <header> = #800001000 (This is an example header)
' The **8** may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block. The
' size can vary depending on the number of points acquired for the
' waveform. You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

' Unsigned integer bytes.
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)
For lngI = 0 To UBound(varQueryResult) - 20
    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 + lngYOrigin) + " V, " + FormatNumber(((lngI / intBytesPerData - lngXReference) * dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
Next lngI
MsgBox "Waveform data:" + vbCrLf + strOutput

' 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

Private Sub CheckForInstrumentErrors()

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
Index

channel probe ID, 207
channel protection, 181
channel reset conditions, 89
channel selected to produce trigger, 390, 429
cell signal type, 180
cell skew for oscilloscope probe, 179, 499
cell status, 134, 503
cell threshold, 498
cell vernier, 185
cell, stop displaying, 109
cell to oscilloscope, 108
cells, how autoscale affects, 105
characters to display, 314
classes of input signals, 223
classifications, command, 564
clear, 195
clear bus command, 155
CLEAR commands, 472
clear cumulative edge variables, 496
clear display, 110
clear markers, 250, 512
clear measurement, 250, 512
clear message queue, 79
clear screen, 504
clear status, 79
clear clipped waveform area, 193
clear clipped high waveform data value, 446
clear clipped low waveform data value, 446
clear clock, 395, 417, 418, 422
CLEAR commands, 472
CLS (Clear Status), 79
CMOS (Command Error) status bit, 81, 83
CMOS threshold voltage for digital channels, 192, 498
CMOS trigger threshold voltage, 531
code, *RST, 91
code, ACQuire:COMPlete, 141
code, ACQuire:TYPE, 149
code, Autoscale, 106
code, CHAnnel:LABel, 175
code, CHAnnel:PROBe, 177
code, CHAnnel:RANGE, 182
code, DIGitize, 111
code, DISPlay:DATA, 197
code, DISPlay:LABel, 198
code, DISPlay:ORDER, 503
code, MEASure:PERiod, 270
code, MEASure:TEGDg, 273
code, POD:THReshold, 290
code, RUN://STOP, 131
code, SYStem:SEtup, 317
code, TIMebase:DElay, 527
code, TIMebase:MODE, 322
code, TIMebase:RANGE, 324
code, TIMebase:REFERENCE, 326
code, TRIGger:MODE, 338
code, TRIGger:SLOPe, 368
code, TRIGger:SOURce, 389
code, VIEW and BLANK, 137
code, WAVEform, 457
code, WAVEform:DATA, 446
code, WAVEform:POINts, 450
code, WAVEform:PREamble, 454
code, VISA COM library example in C, 586
code, VISA library example in Visual Basic, 614
code, VISA library example in C, 595
code, VISA library example in Visual Basic, 604
colon, root commands prefixed by, 102
color palette for hardcopy, 230
Comma Separated Values (CSV) hardcopy format, 228
command classifications, 564
command differences from 54620/54640 series oscilloscopes, 26
command errors detected in Standard Event Status, 83
command header, 566
command headers, common, 568
command headers, compound, 567
command headers, simple, 567
command strings, valid, 565
command tree, 569
commands by subsystem, 395/5/54640 series oscilloscopes, 26
commands, obsolete and discontinued, 446
commands, quick reference, 110
commands, sent over interface, 142
commands, more about, 563
commands, sent over interface, 77
commands, more about, 563
commands, obsolete and discontinued, 491
common (*) commands, 74, 75, 77
common command header, 567
completion criteria for an acquisition, 141, 142
compound command header, 567
compound header, 579
computer control example, 586, 595, 604, 614
conditions for external trigger, 203
conditions, reset, 89
configurations, oscilloscope, 85, 88, 92, 317
connect sampled data points, 502
constants for making automatic measurements, 177
constants for scaling display factors, 177
constants for setting trigger levels, 177
copy display, 130
core commands, 564
count, 408, 445
Nth edge of burst, 361
COUNT commands, 472
count values, 142
counter, 251
coupling, 365
COPUling commands, 473
coupling for channels, 171
CSV (Comma Separated Values) hardcopy format, 228
cumulative edge activity, 496
current digital levels on digital channels, 103
current oscilloscope configuration, 85, 88, 92, 317
current probe, 184, 211
cursor mode, 234
cursor position, 235, 237, 239, 240, 242
cursor readout, 513, 517, 518
cursor reset conditions, 89
cursor source, 236, 238
cursor time, 513, 517, 518
cursors, track measurements, 269
cursors, how autoscale affects, 105
cursors, X1, X2, Y1, Y2, 233
cycle base, FLEXray time trigger, 377
cycle count base, FLEXray frame trigger, 373
cycle count repetition, FLEXray frame trigger, 374
cycle measured, 257
cycle repetition, FLEXray time trigger, 378
cycle time, 263

D
D- source, 432
D+ source, 433
data, 345, 393, 396, 420, 423, 446
data 2, 394
DATA commands, 473
data conversion, 438
data displayed, 196
data format for transfer, 438
data output format, 444
data pattern length, 346
data pattern width, 421
data point index, 465
data points, 145
data required to fill time buckets, 141
data structures, status reporting, 543
data transfer, 196
data, erasing, 110
data, saving and recalling, 193
DATE commands, 473
date, calibration, 160
date, system, 313
dB versus frequency, 213
DC coupling for edge trigger, 365
DC input coupling for specified channel, 171
dc RMS measured on waveform, 282
DDE (Device Dependent Error) status bit, 81, 83
decision chart, status reporting, 561
default conditions, 89
define channel labels, 175
define glitch trigger, 388
define logic thresholds, 498
define measurement, 253
define measurement source, 270
define trigger, 340, 355, 356, 357, 359, 389, 410
defined as, 70
definite-length block response data, 71
definition commands, 473
definition commands, 473
delay measured to calculate phase, 264
delay measurement, 253
delay measurements, 272
delay parameters for measurement, 255
delay, how autoscale affects, 105
delayed time base, 322, 527
delayed time base mode, how autoscale affects, 105
delayed window horizontal scale, 331
da, 513
delta time, 513
delta voltage measurement, 522
delta X cursor, 233
delta Y cursor, 233
DeskJet, 509
destination, 201
detecting probe types, 501, 506
device for hardcopy, 509
device-defined error queue clear, 79
differences from 54820/54840 series oscilloscope commands, 26
differential signal type, 180, 208
differentiate math function, 143, 213, 217, 456
Diff erentiate source for function, 221
digital channel commands, 186, 188, 189, 190, 192
digital channel data, 438
digital channel labels, 199
digital channel order, 503
digital channel source for glitch trigger, 390
digital channels, 4
digital channels, activity and logic levels on, 103
digital channels, groups of, 287, 288, 290
Digitals commands, 186
digital pod, stop displaying, 109
digital reset conditions, 89
digitize channels, 111
digits, 71
disable anti-alias mode, 143
disable front panel, 316
disable function, 507
disabling calibration, 164
disabling channel display, 172
disabling status register bits, 80, 93
discontinued and obsolete commands, 491
display channel labels, 198
display clear, 195
DISPLAY commands, 193, 473
display commands introduction, 193
display connect, 502
display data, 196
display date, 313
display factors scaling, 177
display for channels, 172
display frequency span, 222
display measurements, 248, 249
display order, 503
display persistence, 200
display reference, 323, 326
display reset conditions, 89
display serial number, 132
display source, 201
display vectors, 202
display wave position, 503
display, clearing, 110
display, oscilloscope, 188, 200, 201, 202, 215, 288, 314
display, serial decode bus, 303
displaying a baseline, 342
displaying unsynchronized signal, 342
driver, printer, 231
DSO models, 4
duplicate mnemonics, 579
duration, 355, 356, 359
duration for glitch trigger, 384, 385, 389
duration pattern, 357
duration qualifier, trigger, 355, 356, 358
DURation trigger commands, 354
duration triggering, 332
duty cycle measurement, 248, 257
duty cycle measurement

e, 342
e, 231
duplicate mnemonics, 579
duration, 355, 356, 359
duration for glitch trigger, 384, 385, 389
duration pattern, 357
duration qualifier, trigger, 355, 356, 358
DURation trigger commands, 354
duration triggering, 332
duty cycle measurement, 248, 257

EBURst trigger commands, 360
ECL channel threshold, 498
ECL threshold voltage for digital channels, 192
ECL trigger threshold voltage, 531
digit, 410
edge activity, 496
digitize channels, 111

display frequencies, 509
display, oscilloscope, 188, 200, 201, 202, 215, 288, 314
display, serial decode bus, 303
displaying a baseline, 342
displaying unsynchronized signal, 342
driver, printer, 231
DSO models, 4
duplicate mnemonics, 579
duration, 355, 356, 359
duration for glitch trigger, 384, 385, 389
duration pattern, 357
duration qualifier, trigger, 355, 356, 358
DURation trigger commands, 354
duration triggering, 332
duty cycle measurement, 248, 257

EBURT trigger commands, 360
ECL channel threshold, 498
ECL threshold voltage for digital channels, 192
ECL trigger threshold voltage, 531
digit, 410
edge activity, 496
digitize channels, 111

ECL trigger commands, 364
edge triggering, 332
edges (activity) on digital channels, 103
elements in measurement, 321
enable channel labels, 198
enabling calibration, 164
enabling channel display, 172
enabling status register bits, 80, 93
end of channel (EOC) terminator, 566
end of text (EDT) terminator, 566
end or identify (EOI), 566
enter pattern, 340
EOI (end or identify), 566
EOC (end of channel) terminator, 566
EOC (end of text) terminator, 566
Epson, 509
equivalent-time acquisition mode, 139, 144
erase data, 110, 195
erase functions, 110
erase measurements, 512
erase screen, 504
ERRor commands, 474
error frame count (CAN), 298
error messages, 315, 533
error number, 315
error queue, 315, 554
error, measurement, 248
ESB (Event Status Bit), 94, 96
ESE (Standard Event Status Enable Register), 80, 553
ESR (Standard Event Status Register), 82, 552
event status conditions occurred, 96
Event Status Enable Register (ESE), 80, 553
Event Status Register (ESR), 82, 136, 552
example code, *RST, 91
eexample code, ACQuiPoll:COMplete, 141
eexample code, ACQuiPoll:TYPE, 149
eexample code, Autoscale, 106
eexample code, CHANnel:LABel, 175
eexample code, CHANnel:PROBe, 177
eexample code, CHANnel:RANGE, 182
eexample code, DIGitize, 111
eexample code, DISPlay:DATA, 197
eexample code, DISPlay:LABel, 198
eexample code, DISPlay:ORDER, 503
eexample code, MEASure:PERiod, 270
eexample code, MEASure:TEDGe, 273
eexample code, POD:THReshold, 290
eexample code, RUN:STOP, 131
eexample code, SYSTem:SETup, 317
eexample code, TIMebase:DELay, 527
eexample code, TIMebase:MODE, 322
eexample code, TIMebase:RANGE, 324
eexample code, TIMebase:REFERENCE, 326
eexample code, TRIGger:MODE, 338
eexample code, TRIGger:SLOPe, 368
eexample code, TRIGger:SOURce, 369
eexample code, VIEW and BLANK, 137
eexample code, WAVeform, 457
eexample code, WAVeform:DATA, 446
eexample code, WAVeform:POINts, 450
eexample code, WAVeform:PREamble, 454
example programs, 4, 585, 586, 595, 604, 614
EXE (Execution Error) status bit, 81, 83
execution error detected in Standard Event Status, 83
exponential notation, 70
external glitch trigger source, 390
external range, 210
external trigger, 203, 205, 206, 369, 505
EXTERNAL trigger commands, 203
external trigger input impedance, 205, 505
EXTERNAL trigger level, 366
external trigger overload, 209
external trigger probe attenuation factor, 206
external trigger probe ID, 207
external trigger probe sensing, 506
external trigger protection, 209
external trigger signal type, 208
EXTERNAL trigger source, 369
external trigger units, 211

F
failure, self test, 98
fall time measurement, 248, 258
falling edge, 340, 410
Fast Fourier Transform (FFT) functions, 213, 214, 219, 221, 222, 223
FFT values in waveform data, 446
FFT (Fast Fourier Transform) functions, 213, 214, 219, 221, 222, 223
introduction to :SBUS commands, 293
introduction to :SYSTem commands, 312
introduction to :TIMebase commands, 320
introduction to :TRIGger commands, 332
introduction to :WAVeform commands, 438
introduction to common (*) commands, 77
introduction to root () commands, 102
invert gratucile colors for hardcopy, 220
inverting input for channels, 174

K
key disable, 316
key press detected in Standard Event Status Register, 83
knob disable, 316
known state, 89

L
label, 189, 497
label command, bus, 157
LABel commands, 477
label list, 175, 199
labels, 175, 198, 199
labels to store calibration information, 161
labels, specifying, 193
language example program, 586, 595, 604, 614
LaserJet, 509
leakage into peak spectrum, 223
learn string, 85, 317
least significant byte first, 444
left reference, 326
legal values for channel offset, 176
legal values for frequency span, 222
legal values for offset, 216
less than qualifier, 388
less than time, 356, 358, 385, 389
LESSthan commands, 477
LEVEL commands, 477
level for trigger voltage, 366, 386
LF coupling, 365
license information, 87
limits for line number, 426
LIN acknowledge, 402
LIN baud rate, 403
LIN identifier, 401
LIN serial decode bus parity bits, 309
LIN source, 404
LIN standard, 405
LIN sync break, 406
LIN trigger, 407
LIN trigger commands, 400
LIN trigger definition, 530
LIN triggering, 332
line glitch trigger source, 390
line number for TV trigger, 426
line terminator, 70
LINE trigger level, 366
LINE trigger source, 369
list of channel labels, 199
load utilization (CAN), 302
local lockout, 316
lock, 316
lockout message, 316
logic level activity, 496
long form, 566
lower threshold, 263
lower threshold voltage for measurement, 511
lowercase characters in commands, 565
low-frequency reject filter, 367
low-pass filter used to limit bandwidth, 170, 204
LRN (Learn Device Setup), 85
lsbfirst, 444

M
magnitude of occurrence, 274
main sweep range, 329
main time base mode, 322
making measurements, 248
MAN option for probe sense, 501, 506
manual cursor mode, 234
MARKer commands, 232
marker mode, 240
marker position, 241
marker readout, 517, 518
marker set for voltage measurement, 523, 524
marker sets start time, 514
marker time, 513
markers for delta voltage measurement, 522
markers track measurements, 269
markers, command overview, 233
markers, mode, 234
markers, time at start, 518
markers, time at stop, 517
markers, X delta, 239
markers, X1 position, 235
markers, X1Y1 source, 236
markers, X2 position, 237
markers, X2Y2 source, 238
markers, Y delta, 242
markers, Y1 position, 240
markers, Y2 position, 241
mask, 80, 93, 340, 357
mask command, bus, 158
master summary status bit, 96
math function, stop displaying, 109
math operations, 213
MAV (Message Available), 79, 94, 96
maximum duration, 355, 356, 385
maximum position, 323
maximum range for delayed window, 330
maximum scale for delayed window, 331
maximum vertical value measurement, 279
maximum vertical value, time of, 285, 515
MEASure commands, 243
measure overshoot, 261
measure period, 263
measure phase between channels, 264
measure preshoot, 265
measure start voltage, 523
measure stop voltage, 524
measure value at a specified time, 283
measure value at top of waveform, 284
measurement error, 248
measurement setup, 248, 270
measurement source, 270
measurements, average value, 277
measurements, base value, 278
measurements, clear, 250, 512
measurements, command overview, 248
measurements, counter, 251
measurements, dc RMS, 282
measurements, definition setup, 253
measurements, delay, 255
measurements, duty cycle, 257
measurements, fall time, 258
measurements, frequency, 259
measurements, how autoscale affects, 105
measurements, lower threshold level, 511
measurements, maximum vertical value, 279
measurements, maximum vertical value, time of, 285, 515
measurements, minimum vertical value, 280
measurements, minimum vertical value, time of, 286, 516
measurements, overshoot, 261
measurements, period, 263
measurements, phase, 264
measurements, preshoot, 265
measurements, pulse width, negative, 260
measurements, pulse width, positive, 266
measurements, resettting, 110
measurements, rise time, 267
measurements, show, 269
measurements, source channel, 270
measurements, standard deviation, 268
measurements, start marker time, 517
measurements, stop marker time, 518
measurements, thresholds, 514
measurements, time between start and stop markers, 513
measurements, time between trigger and edge, 272
measurements, time between trigger and vertical value, 274
measurements, time between trigger and voltage level, 519
measurements, upper threshold value, 521
measurements, vertical amplitude, 276
measurements, vertical peak-to-peak, 281
measurements, voltage difference, 522
memory setup, 92, 317
merge, 119
message available bit, 96
message available bit clear, 79
message displayed, 96
message error, 533
message queue, 551
messages ready, 96
midpoint of thresholds, 263
minimum duration, 355, 356, 359, 384
minimum vertical value measurement, 280
minimum vertical value, time of, 286, 516
mixed-signal oscilloscopes, 4
mnemonics, duplicate, 579
mode, 144, 148, 234, 322, 427
MODE commands, 479
mode, serial decode, 310
model number, 84
models, oscilloscope, 3
modes for triggering, 338
most significant byte first, 444
move, 213
move cursors, 517, 518
msbfirst, 444
MSG (Message), 94, 96
MSD models, 4
MSS (Master Summary Status), 96
multiple commands, 579
multiply math function, 213, 217, 456

N
name channels, 175
name list, 199
negative glitch trigger polarity, 387
negative pulse width, 260
negative slope, 368, 417
Nth edge in burst, 363
negative TV trigger polarity, 428
new line (NL) terminator, 70, 566
NL (new line) terminator, 70, 566
noise reject filter, 339
non-core commands, 564
non-interlaced GEneric mode, 430
non-volatile memory, label list, 157, 189, 199
normal acquisition type, 138, 438
normal trigger sweep mode, 332
notices, 2
NR1 number format, 70
NR3 number format, 70
Nth edge in a burst idle, 362
Nth edge in burst slope, 363
Nth edge of burst counter, 361
NTSC, 426, 430
null frame count (FlexRay), 304
NULL string, 314
number format, 70
number of points, 145, 449, 451
time of number of buckets, 448, 451

width, 260

O
obsolete and discontinued commands, 491
obsolete commands, 564
occurrence reported by magnitude, 519
offset, 213
OFFSet commands, 479
offset value for channel voltage, 176
offset value for selected function, 216
one values in waveform data, 446
OPC (Operation Complete) command, 86
OPC (Operation Complete) status bit, 81, 83
OPEE (Operation Status Enable Register), 120
operating configuration, 85, 317
operating state, 92
operation complete, 86
operation status condition register, 122
Operation Status Condition Register
`:OPERegister:CONDition`, 122, 556
operation status conditions occurred, 96
Operation Status Enable Register (OPEE), 120
operation status event register, 124
Operation Status Event Register
`:OPERegister:EVENt`, 124, 555
operation, math, 213
operations for function, 217
OPERegister:CONDition (Operation Status Condition Register), 122, 556
OPERegister:EVENt (Operation Status Event Register), 124, 555
OPT (Option Identification), 87
optional syntax terms, 70
options, 87
order of digital channels on display, 503
order of output, 444
oscilloscope external trigger, 203
oscilloscope models, 3
oscilloscope rate, 147
output messages ready, 96
output queue, 86, 550
output queue clear, 79
output sequence, 444
overlapped commands, 583
overload, 181, 209
Overload Event Enable Register (OVL), 126
Overload Event Register (OVL), 128
overload frame count (CAN), 299
overload protection, 126, 128
overshoot of waveform, 261
overvoltage, 181, 209
OVL (Overload Event Enable Register), 126
OVL (Overload Event Register), 128
OVL bit, 117, 122, 124

P
P1080L24HZ, 426, 430
P1080L25HZ, 426, 430
P480L60HZ, 426, 430
P720L60HZ, 426, 430
PAL, 426, 430
palette for hardcopy, 230
PAL-M, 426, 430
parameters for delay measurement, 255
parametric measurements, 248
parity bits, LIN serial decode bus, 309
parser, 102, 580
pass, self test, 98
pattern, 340, 345, 347, 357, 392, 393, 394,
412, 420
pattern and edge, 340
PATtern commands, 480
pattern duration, 355, 356, 384, 385
pattern length, 346
pattern trigger, 340
pattern triggering, 332
pattern width, 421
peak detect, 148
peak detect acquisition type, 139, 438
peaks, 213
peak-to-peak vertical value measurement, 281
pending operations, 88
percent of waveform overshoot, 261
percent thresholds, 253
period measured to calculate phase, 264
period measurement, 248, 263
persistance, waveform, 193, 200
phase measured between channels, 264
phase measurements, 272
pixel memory, 201
pixel memory, saving display to, 119
PLL Locked bit, 115, 122
pod, 287, 288, 289, 290, 456, 498
P0D commands, 287
pod, stop displaying, 109
points, 145, 449, 451
POI Nts commands, 480
points in waveform data, 438
polarity, 428
POLarity commands, 480
polarity for glitch trigger, 387
PDN (Power On) status bit, 81, 83
position, 190, 237, 323, 329
POSitioN commands, 480
position cursors, 517, 518
position in delayed view, 329
position waveforms, 503
positive glitch trigger polarity, 387
positive pulse width, 266
positive slope, 368, 417
Nth edge in burst, 363
positive TV trigger polarity, 428
positive width, 260
preamble data, 438, 453
predefined logic threshold, 498
predefined threshold voltages, 531
preset conditions, 89
preshoot measured on waveform, 265
previously stored configuration, 88
print command, 130
print query, 525
printer, 509
printer driver for hardcopy, 231
printer hardcopy format, 228
printing, 224
printing in grayscale, 510
probe, 366
probe attenuation affects channel voltage range, 182
probe attenuation factor (external trigger), 206
probe attenuation factor for selected channel, 177
PROBe commands, 481
probe ID, 178, 207
program data, 566
program data syntax rules, 568
program example, 586, 595, 604, 614
program message, 77
program message syntax, 565
program message terminator, 566
programming examples, 4, 585
protecting against calibration, 164
protection, 126, 128, 181, 209
PRoTection commands, 481
pulse width, 260, 266
pulse width duration trigger, 384, 385, 389
pulse width measurement, 248
pulse width trigger, 339
pulse width trigger level, 386
pulse width triggering, 332
pwidth, 266

Q
qualifier, 389
QUALifier commands, 481
qualifier, trigger duration, 355, 356, 358
query error detected in Standard Event
Status, 83
query return values, 582
query setup, 224, 233, 248, 317
query subsystem, 151, 186, 213
querying setup, 168
querying the subsystem, 332
queues, cleaning, 580
quick reference, commands, 31
quoted ASCII string, 71
QYE (Query Error) status bit, 81, 83

R
range, 213, 330
RANGe commands, 481
range for channels, 182
range for duration trigger, 359
range for external trigger, 210
range for full-scale vertical axis, 218
range for glitch trigger, 389
range for time base, 324
range of offset values, 176
range of reference level values, 219
range qualifier, 388
ranges, value, 71
rate, 147
RCL (Recall), 88
read configuration, 85
read trace memory, 196
readout, 513
real-time acquisition mode, 139, 144
recall, 88, 317
recalling and saving data, 193
RECtangular window for transient signals, 223
reference, 213, 326
reference clock, 325
REFerence commands, 481
reference for time base, 527
reference level, Fast Fourier Transform (FFT) function, 219
reference signal (10 MHz), 146
reference signal mode, 325
registers, 82, 88, 92, 104, 113, 115, 117, 120, 122, 124, 126, 128
registers, clearing, 560
reject filter, 367
reject high frequency, 336
reject noise, 339
remote control example, 586, 595, 604, 614
remove cursor information, 234
remove labels, 198
remove message from display, 314
reorder channels, 105
repetitive acquisitions, 131
report errors, 315
report transition, 272, 274
reporting status, 541
reporting the setup, 332
request service, 96
Request-for-OPC flag clear, 79
reset, 89, 413
RESet commands, 481
reset conditions, 89
reset measurements, 110, 195
resolution of printed copy, 510
restore configurations, 85, 88, 92, 317
restore labels, 198
restore setup, 88
return values, query, 582
returning acquisition type, 148
returning number of data points, 145
right reference, 326
rise time measurement, 248
rise time of positive edge, 267
rising edge, 340, 410
RMS value measurement, 282
roll time base mode, 322
root (:) commands, 100, 102
root level commands, 74
RQL (Request Control) status bit, 81, 83
ROS (Request Service), 96
RST (Reset), 89
rules, tree traversal, 579
rules, truncation, 566
run, 97, 131
Run bit, 122, 124
running configuration, 92, 317
save and recalling data, 193
S
sample rate, 147
sampled data, 502
sampled data points, 446
SAV (Save), 92
save, 92
SAVE TO INTERN, 119
save waveforms to pixel memory, 119
scale, 220, 327, 331
SCALEs commands, 482
scale factors output on hardcopy, 225
scale for channels, 183
scale units for channels, 184
scale units for external trigger, 211
scaling display factors, 177
scratch measurements, 512
screen data, 196
SECAM, 426, 430
seconds per division, 327
segment, FLEXray time trigger, 379
select measurement channel, 270
self-test, 98
sensing a channel probe, 501
sensing a external trigger probe, 506
sensitivity of oscilloscope input, 177
sequence, 411, 412, 413
sequence trigger, 415
SEQUence trigger commands, 408
sequence triggering, 332
sequencer edge counter, 409
sequencer timer, 414
sequential commands, 583
serial clock, 399, 422
serial data, 396, 423
serial decode bus, 293
serial decode bus display, 303
serial decode mode, 310
serial frame, 424
serial number, 132
service request, 96
Service Request Enable Register (SRE), 94, 548
set, 89
set center frequency, 214
set conditions, 105
set cursors, 517, 518
set date, 313
set delay, 105
set thresholds, 105
set time, 319
set time/div, 105
setting digital display, 188
setting digital label, 157, 189
setting digital position, 190
setting digital thresholds, 192
setting display, 215
setting external trigger level, 203
setting impedance for channels, 173
setting inversion for channels, 174
setting pod display, 288
setting pod size, 289
setting pod threshold, 290
settings, 88, 92
settings, instrument, 224
setup, 139, 151, 168, 166, 193, 213, 224, 317
setup configuration, 88, 92, 317
setup defaults, 89
setup memory, 88
setup reported, 332
short form, 3, 566
show channel labels, 198
show measurements, 248, 269
SICL example in C, 586
SIGNal commands, 483
signal type, 180, 208
simple command headers, 567
single acquisition, 133
single-ended signal type, 180, 208
size, 191, 289
SIZE commands, 483
skew, 179, 499
slope, 368, 417
Nth edge in burst, 363
slope (direction) of waveform, 519
SLOPe commands, 483
slope not valid in TV trigger mode, 368
slope of edge, 410
slope parameter for delay measurement, 255
software version, 84
source, 201, 213, 270, 351, 404, 456
SOURCe commands, 483
source for function, 221
source for trigger, 368
span, 213
span of frequency on display, 222
specify measurement, 270
SPI, 417, 418, 420
SPI commands, 484
SPI decode word width, 311
SPI trigger, 419, 421
SPI trigger clock, 422
SPI trigger commands, 416
SPI trigger data, 423
SPI trigger frame, 424
SPI triggering, 332
SRE (Service Request Enable Register), 94, 548
SRQ (Service Request interrupt), 113, 120
standard deviation measured on waveform, 268
Standard Event Status Enable Register (ESE), 80, 553
Standard Event Status Register (ESR), 82, 552
standard for video, 430
standard, LIN, 405
start acquisition, 135
start cursor, 518
stop acquisition, 135
stop cursor, 518
stop displaying channel, 109
stop displaying math function, 109
stop displaying pod, 109
stop time, 389, 518
storage, 92
store instrument setup, 85, 92
store setup, 92
store waveforms to pixel memory, 119
storing calibration information, 161
string, quoted ASCII, 71
subsystem commands, 74, 580
subtract math function, 213, 217, 456
sweep mode, trigger, 332, 342
sweep speed set to fast to measure fall time, 258
sweep speed set to fast to measure rise time, 267
switch disable, 316
switch, calibration protect, 164
sync break, LIN, 406
sync frame count (FlexRay), 306
syntax elements, 70
syntax rules, program data, 568
syntax, optional terms, 70
syntax, program message, 565
SYSTem commands, 312
system commands, 313, 314, 315, 316, 317, 319
system commands introduction, 312
T
tdelta, 513
tedge, 272
telnet ports 5024 and 5025, 446
temporary message, 314
TER (Trigger Event Register), 136, 549
test, self, 98
text, writing to display, 314
threshold, 192, 290, 498, 531
THReshold commands, 484
temperature voltage (lower) for measurement, 511
threshold voltage (upper) for measurement, 521
thresholds, 253, 514
thresholds used to measure period, 263
thresholds, how autoscale affects, 105
time base, 322, 323, 324, 326, 327, 527
time base commands introduction, 320
time base reset conditions, 89
time base window, 329, 330, 331
time between points, 513
time buckets, 141, 142
TIME commands, 484
time delay, 527
time delta, 513
time difference between data points, 463
time duration, 355, 356, 359, 389
time holdoff for trigger, 337
time interval, 272, 274, 513
time interval between trigger and occurrence, 519
time marker sets start time, 514
time per division, 324
time record, 223
time slot, FLEXRay time trigger, 380
time specified, 283
time, calibration, 166
time, start marker, 517
time, stop marker, 518
time, system, 319
time/div, how autoscale affects, 105
time-at-max measurement, 515
time-at-min measurement, 516
TImebase commands, 320
timebase vernier, 328
time-ordered label list, 199
timeout, 418
timer, 414
timing measurement, 248
title channels, 175
top of waveform value measured, 284
TOtal commands, 485
total frame count (CAN), 301
total frame count (FlexRay), 307
trace memories, how autoscale affects, 105
trace memory, 134, 137
trace memory data, 196
track measurements, 269
trademarks, 2
transfer instrument state, 85, 317
transmit, 196
tree traversal rules, 579
tree, command, 569
TRG (Trigger), 94, 96, 97
trigger (external) input impedance, 205, 505
trigger armed event register, 122, 124
TRIGger CAN commands, 343
trigger channel source, 390, 429
TRIGger commands, 332, 485
TRIGger commands, general, 335
trigger duration, 355, 356
TRIGger DURation commands, 354
TRIGger EBUsRt commands, 360
trigger edge, 410
TRIGger EDGE commands, 364
trigger edge coupling, 365
trigger edge slope, 368
trigger event bit, 136
Trigger Event Register (TER), 549
TRIGger FLEXRay commands, 370
TRIGger GLitch commands, 382
trigger holdoff, 337
TRIGger IIC commands, 391
trigger occurred, 96
trigger pattern, 340, 357
trigger qualifier, 358
trigger reset conditions, 89
Index

voltage offset value for channels, 176
voltage probe, 184, 211
voltage ranges for channels, 182
voltage ranges for external trigger, 210
voltage threshold, 253

W
WAI (Wait To Continue), 99
wait, 99
wait for operation complete, 86
Wait Trig bit, 122, 124
waveform base value measured, 278
Waveform commands, 436
waveform data, 438
waveform introduction, 438
waveform maximum vertical value measured, 279
waveform minimum vertical value measured, 280
waveform must cross voltage level to be an occurrence, 519
waveform peak-to-peak vertical value measured, 281
waveform period, 263
waveform persistence, 193
waveform RMS value measured, 282
waveform source channels, 456
waveform standard deviation value measured, 268
waveform vertical amplitude, 276
waveform voltage measured at marker, 523, 524
waveform, byte order, 444
waveform, count, 445
waveform, data, 446
waveform, format, 448
waveform, points, 449, 451
waveform, preamble, 453
waveform, source, 456
waveform, type, 460
waveform, unsigned, 461
waveform, view, 462
waveform, X increment, 463
waveform, X origin, 464
waveform, X reference, 465
waveform, Y increment, 466
waveform, Y origin, 467
waveform, Y reference, 468
what’s new, 17
width, 389
WIDTH commands, 489
window, 329, 330, 331
window time, 324
window time base mode, 322
windows, 223
windows as filters to Fast Fourier Transforms, 223
windows for Fast Fourier Transform functions, 223
word format, 448
word format for data transfer, 438

word width, SPI decode, 311
write text to display, 314
write trace memory, 196

X
X axis markers, 233
X delta, 239
X1 and X2 cursor value difference, 239
X1 cursor, 233, 235, 236
X2 cursor, 233, 237, 238
X-axis functions, 320
X-increment, 463
X-of-max measurement, 285
X-of-min measurement, 286
X-origin, 464
X-reference, 465
X-Y mode, 320, 322

Y
Y axis markers, 233
Y1 and Y2 cursor value difference, 242
Y1 cursor, 233, 236, 240, 242
Y2 cursor, 233, 238, 241, 242
Y-axis value, 467
Y-increment, 466
Y-origin, 467, 468
Y-reference, 468

Z
zero values in waveform data, 446