Agilent InfiniiVision 5000 Series Oscilloscopes

Programmer's Reference

Agilent Technologies

Notices

Trademarks

Manual Part Number

Version 05.00.0001

Edition

January 28, 2008

Available in electronic format only

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

Warranty

Technology Licenses

Restricted Rights Legend

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 InfiniiVision 5000 Series Oscilloscope Models

Channels	**Input Bandwidth (Maximum Sample Rate)**		
	500 MHz (4 GSa/s)	**300 MHz (2 GSa/s)**	**100 MHz (2 GSa/s)**
4 analog	DSO5054A	DSO5034A	DSO5014A
2 analog	DSO5052A	DSO5032A	DSO5012A

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 23
- "Commands by Subsystem" on page 59
- "Commands A-Z" on page 435
- "Obsolete and Discontinued Commands" on page 457
- "Error Messages" on page 499
- "Status Reporting" on page 507
- "More About Oscilloscope Commands" on page 529
- "Programming Examples" on page 549

See the *Agilent InfiniiVision 5000 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.

Example Programs

The example programs are designed to work with multiple InfiniiVision 5000 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 5.00 18

What's New in Version 4.10 20

Version 4.00 at Introduction 21

2 Commands Quick Reference

Command Summary 24

Syntax Elements 56
- Number Format 56
- <NL> (Line Terminator) 56
- [ ] (Optional Syntax Terms) 56
- { } (Braces) 56
- ::= (Defined As) 56
- < > (Angle Brackets) 57
- ... (Ellipsis) 57
- n,..,p (Value Ranges) 57
- d (Digits) 57
- Quoted ASCII String 57
- Definite-Length Block Response Data 57

3 Commands by Subsystem

Common (*) Commands 61
- *CLS (Clear Status) 65
- *ESE (Standard Event Status Enable) 66
- *ESR (Standard Event Status Register) 68
- *IDN (Identification Number) 70
- *LRN (Learn Device Setup) 71
- *OPC (Operation Complete) 72
- *OPT (Option Identification) 73
- *RCL (Recall) 74
- *RST (Reset) 75
- *SAV (Save) 78

*SRE (Service Request Enable) 79
*STB (Read Status Byte) 81
*TRG (Trigger) 83
*TST (Self Test) 84
*WAI (Wait To Continue) 85

Root (:) Commands 86
:AER (Arm Event Register) 89
:AUToscale 90
:AUToscale:AMODE 92
:AUToscale:CHANnels 93
:BLANk 94
:CDISplay 95
:DIGitize 96
:HWEenable (Hardware Event Enable Register) 98
:HWERegister:CONDition (Hardware Event Condition Register) 100
:HWERegister[:EVENt] (Hardware Event Event Register) 102
:MERGe 104
:OPEE (Operation Status Enable Register) 105
:OPERegister:CONDition (Operation Status Condition Register) 107
:OPERegister[:EVENt] (Operation Status Event Register) 109
:OVLenable (Overload Event Enable Register) 111
:OVLRegister (Overload Event Register) 113
:PRINt 115
:RUN 116
:SERial 117
:SINGle 118
:STATus 119
:STOP 120
:TER (Trigger Event Register) 121
:VIEW 122

:ACQuire Commands 123
:ACQuire:AALias 125
:ACQuire:COMPlete 126
:ACQuire:COUNt 127
:ACQuire:DAALias 128
:ACQuire:MODE 129
:ACQuire:POINts 130
:ACQuire:SRATe 131
:ACQuire:TYPE 132

:CALibrate Commands 134

:CALibrate:DATE 135
:CALibrate:LABel 136
:CALibrate:STARt 137
:CALibrate:STATus 138
:CALibrate:SWITch 139
:CALibrate:TEMPerature 140
:CALibrate:TIME 141

:CHANnel<n> Commands 142
:CHANnel<n>:BWLimit 145
:CHANnel<n>:COUPling 146
:CHANnel<n>:DISPlay 147
:CHANnel<n>:IMPedance 148
:CHANnel<n>:INVert 149
:CHANnel<n>:LABel 150
:CHANnel<n>:OFFSet 151
:CHANnel<n>:PROBe 152
:CHANnel<n>:PROBe:ID 153
:CHANnel<n>:PROBe:SKEW 154
:CHANnel<n>:PROBe:STYPe 155
:CHANnel<n>:PROTection 156
:CHANnel<n>:RANGe 157
:CHANnel<n>:SCALe 158
:CHANnel<n>:UNITs 159
:CHANnel<n>:VERNier 160

:DISPlay Commands 161
:DISPlay:CLEar 163
:DISPlay:DATA 164
:DISPlay:LABel 166
:DISPlay:LABList 167
:DISPlay:PERSistence 168
:DISPlay:SOURce 169
:DISPlay:VECTors 170

:EXTernal Trigger Commands 171
:EXTernal:BWLimit 173
:EXTernal:IMPedance 174
:EXTernal:PROBe 175
:EXTernal:PROBe:ID 176
:EXTernal:PROBe:STYPe 177
:EXTernal:PROTection 178
:EXTernal:RANGe 179

:EXTernal:UNITs 180

:FUNCtion Commands 181

:FUNCtion:CENTer 183

:FUNCtion:DISPlay 184

:FUNCtion:OFFSet 185

:FUNCtion:OPERation 186

:FUNCtion:RANGe 187

:FUNCtion:REFerence 188

:FUNCtion:SCALe 189

:FUNCtion:SOURce 190

:FUNCtion:SPAN 191

:FUNCtion:WINDow 192

:HARDcopy Commands 193

:HARDcopy:AREA 195

:HARDcopy:APRinter 196

:HARDcopy:FACTors 197

:HARDcopy:FFEed 198

:HARDcopy:INKSaver 199

:HARDcopy:PALette 200

:HARDcopy:PRinter:LIST 201

:HARDcopy:STARt 202

:MARKer Commands 203

:MARKer:MODE 205

:MARKer:X1Position 206

:MARKer:X1Y1source 207

:MARKer:X2Position 208

:MARKer:X2Y2source 209

:MARKer:XDELta 210

:MARKer:Y1Position 211

:MARKer:Y2Position 212

:MARKer:YDELta 213

:MEASure Commands 214

:MEASure:CLEar 220

:MEASure:COUNter 221

:MEASure:DEFine 222

:MEASure:DELay 225

:MEASure:DUTYcycle 227

:MEASure:FALLtime 228

:MEASure:FREQuency 229

:MEASure:NWIDth 230

:MEASure:OVERshoot 231
:MEASure:PERiod 233
:MEASure:PHASe 234
:MEASure:PREShoot 235
:MEASure:PWIDth 236
:MEASure:RISetime 237
:MEASure:SDEViation 238
:MEASure:SHOW 239
:MEASure:SOURce 240
:MEASure:TEDGe 242
:MEASure:TVALue 244
:MEASure:VAMPlitude 246
:MEASure:VAVerage 247
:MEASure:VBASe 248
:MEASure:VMAX 249
:MEASure:VMIN 250
:MEASure:VPP 251
:MEASure:VRMS 252
:MEASure:VTIMe 253
:MEASure:VTOP 254
:MEASure:XMAX 255
:MEASure:XMIN 256

:RECall Commands 257
:RECall:FILename 258
:RECall:IMAGe[:STARt] 259
:RECall:PWD 260
:RECall:SETup[:STARt] 261

:SAVE Commands 262
:SAVE:FILename 264
:SAVE:IMAGe[:STARt] 265
:SAVE:IMAGe:AREA 266
:SAVE:IMAGe:FACTors 267
:SAVE:IMAGe:FORMat 268
:SAVE:IMAGe:INKSaver 269
:SAVE:IMAGe:PALette 270
:SAVE:PWD 271
:SAVE:SETup[:STARt] 272
:SAVE:WAVeform[:STARt] 273
:SAVE:WAVeform:FORMat 274
:SAVE:WAVeform:LENGth 275

:SBUS Commands 276
- :SBUS:CAN:COUNt:ERRor 278
- :SBUS:CAN:COUNt:OVERload 279
- :SBUS:CAN:COUNt:RESet 280
- :SBUS:CAN:COUNt:TOTal 281
- :SBUS:CAN:COUNt:UTILization 282
- :SBUS:DISPlay 283
- :SBUS:IIC:ASIZe 284
- :SBUS:LIN:PARity 285
- :SBUS:MODE 286
- :SBUS:SPI:WIDTh 287
- :SBUS:UART:BASE 288
- :SBUS:UART:COUNt:ERRor 289
- :SBUS:UART:COUNt:RESet 290
- :SBUS:UART:COUNt:RXFRames 291
- :SBUS:UART:COUNt:TXFRames 292
- :SBUS:UART:FRAMing 293

:SYSTem Commands 294
- :SYSTem:DATE 295
- :SYSTem:DSP 296
- :SYSTem:ERRor 297
- :SYSTem:LOCK 298
- :SYSTem:SETup 299
- :SYSTem:TIME 301

:TIMebase Commands 302
- :TIMebase:MODE 304
- :TIMebase:POSition 305
- :TIMebase:RANGe 306
- :TIMebase:REFerence 307
- :TIMebase:SCALe 308
- :TIMebase:VERNier 309
- :TIMebase:WINDow:POSition 310
- :TIMebase:WINDow:RANGe 311
- :TIMebase:WINDow:SCALe 312

:TRIGger Commands 313
- General :TRIGger Commands 316
- :TRIGger:HFReject 317
- :TRIGger:HOLDoff 318
- :TRIGger:MODE 319
- :TRIGger:NREJect 320

:TRIGger:PATTern 321
:TRIGger:SWEep 323
:TRIGger:CAN Commands 324
:TRIGger:CAN:PATTern:DATA 326
:TRIGger:CAN:PATTern:DATA:LENGth 327
:TRIGger:CAN:PATTern:ID 328
:TRIGger:CAN:PATTern:ID:MODE 329
:TRIGger:CAN:SAMPlepoint 330
:TRIGger:CAN:SIGNal:BAUDrate 331
:TRIGger:CAN:SOURce 332
:TRIGger:CAN:TRIGger 333
:TRIGger:DURation Commands 335
:TRIGger:DURation:GREaterthan 336
:TRIGger:DURation:LESSthan 337
:TRIGger:DURation:PATTern 338
:TRIGger:DURation:QUALifier 339
:TRIGger:DURation:RANGe 340
:TRIGger[:EDGE] Commands 341
:TRIGger[:EDGE]:COUPling 342
:TRIGger[:EDGE]:LEVel 343
:TRIGger[:EDGE]:REJect 344
:TRIGger[:EDGE]:SLOPe 345
:TRIGger[:EDGE]:SOURce 346
:TRIGger:GLITch Commands 347
:TRIGger:GLITch:GREaterthan 348
:TRIGger:GLITch:LESSthan 349
:TRIGger:GLITch:LEVel 350
:TRIGger:GLITch:POLarity 351
:TRIGger:GLITch:QUALifier 352
:TRIGger:GLITch:RANGe 353
:TRIGger:GLITch:SOURce 354
:TRIGger:IIC Commands 355
:TRIGger:IIC:PATTern:ADDRess 356
:TRIGger:IIC:PATTern:DATA 357
:TRIGger:IIC:PATTern:DATa2 358
:TRIGger:IIC:SOURce:CLOCk 359
:TRIGger:IIC:SOURce:DATA 360
:TRIGger:IIC:TRIGger:QUALifier 361
:TRIGger:IIC:TRIGger[:TYPE] 362
:TRIGger:LIN Commands 364
:TRIGger:LIN:ID 365
:TRIGger:LIN:SAMPlepoint 366

:TRIGger:LIN:SIGNal:BAUDrate 367
:TRIGger:LIN:SOURce 368
:TRIGger:LIN:STANdard 369
:TRIGger:LIN:SYNCbreak 370
:TRIGger:LIN:TRIGger 371
:TRIGger:SPI Commands 372
:TRIGger:SPI:CLOCk:SLOPe 373
:TRIGger:SPI:CLOCk:TIMeout 374
:TRIGger:SPI:FRAMing 375
:TRIGger:SPI:PATTern:DATA 376
:TRIGger:SPI:PATTern:WIDTh 377
:TRIGger:SPI:SOURce:CLOCk 378
:TRIGger:SPI:SOURce:DATA 379
:TRIGger:SPI:SOURce:FRAMe 380
:TRIGger:TV Commands 381
:TRIGger:TV:LINE 382
:TRIGger:TV:MODE 383
:TRIGger:TV:POLarity 384
:TRIGger:TV:SOURce 385
:TRIGger:TV:STANdard 386
:TRIGger:UART Commands 387
:TRIGger:UART:BAUDrate 389
:TRIGger:UART:BITorder 390
:TRIGger:UART:BURSt 391
:TRIGger:UART:DATA 392
:TRIGger:UART:IDLE 393
:TRIGger:UART:PARity 394
:TRIGger:UART:POLarity 395
:TRIGger:UART:QUALifier 396
:TRIGger:UART:SOURce:RX 397
:TRIGger:UART:SOURce:TX 398
:TRIGger:UART:TYPE 399
:TRIGger:UART:WIDTh 400

:WAVeform Commands 401

:WAVeform:BYTeorder 408
:WAVeform:COUNt 409
:WAVeform:DATA 410
:WAVeform:FORMat 412
:WAVeform:POINts 413
:WAVeform:POINts:MODE 415
:WAVeform:PREamble 417

:WAVeform:SOURce 420
:WAVeform:SOURce:SUBSource 424
:WAVeform:TYPE 425
:WAVeform:UNSigned 426
:WAVeform:VIEW 427
:WAVeform:XINCrement 428
:WAVeform:XORigin 429
:WAVeform:XREFerence 430
:WAVeform:YINCrement 431
:WAVeform:YORigin 432
:WAVeform:YREFerence 433

4 Commands A-Z

5 Obsolete and Discontinued Commands

:CHANnel:LABel 462
:CHANnel2:SKEW 463
:CHANnel<n>:INPut 464
:CHANnel<n>:PMODe 465
:DISPlay:CONNect 466
:ERASe 467
:EXTernal:INPut 468
:EXTernal:PMODe 469
:FUNCtion:VIEW 470
:HARDcopy:DESTination 471
:HARDcopy:DEVice 472
:HARDcopy:FILename 473
:HARDcopy:FORMat 474
:HARDcopy:GRAYscale 475
:HARDcopy:IGColors 476
:HARDcopy:PDRiver 477
:MEASure:LOWer 478
:MEASure:SCRatch 479
:MEASure:TDELta 480
:MEASure:THResholds 481
:MEASure:TMAX 482
:MEASure:TMIN 483
:MEASure:TSTArt 484
:MEASure:TSTOp 485
:MEASure:TVOLt 486
:MEASure:UPPer 488
:MEASure:VDELta 489

:MEASure:VSTArt 490
:MEASure:VSTOp 491
:PRINt? 492
:TIMebase:DELay 494
:TRIGger:CAN:ACKNowledge 495
:TRIGger:CAN:SIGNal:DEFinition 496
:TRIGger:LIN:SIGNal:DEFinition 497
:TRIGger:TV:TVMode 498

6 Error Messages

7 Status Reporting

Status Reporting Data Structures 509

Status Byte Register (STB) 512

Service Request Enable Register (SRE) 514

Trigger Event Register (TER) 515

Output Queue 516

Message Queue 517

(Standard) Event Status Register (ESR) 518

(Standard) Event Status Enable Register (ESE) 519

Error Queue 520

Operation Status Event Register (:OPERegister[:EVENt]) 521

Operation Status Condition Register (:OPERegister:CONDition) 522

Arm Event Register (AER) 523

Hardware Event Event Register (:HWERegister[:EVENt]) 524

Hardware Event Condition Register (:HWERegister:CONDition) 525

Clearing Registers and Queues 526

Status Reporting Decision Chart 527

8 More About Oscilloscope Commands

Command Classifications 530
Core Commands 530
Non-Core Commands 530
Obsolete Commands 530

Valid Command/Query Strings 531

Program Message Syntax 531

Command Tree 535

Duplicate Mnemonics 545

Tree Traversal Rules and Multiple Commands 545

Query Return Values 547

All Oscilloscope Commands Are Sequential 548

9 Programming Examples

SICL Example in C 550

VISA Example in C 559

VISA Example in Visual Basic 568

VISA COM Example in Visual Basic 578

Index

1
What's New

What's New in Version 5.00 18
What's New in Version 4.10 20
Version 4.00 at Introduction 21

What's New in Version 5.00

New features in version 5.00 of the InfiniiVision 5000 Series oscilloscope software are:

- Serial triggering and decode options are now available.
- The :SAVE and :RECall command subsystems.
- Changes to the :HARDcopy sommand subsystem to make a clearer distinction between printing and save/recall functionality.

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

New Commands

Command	Description
:HARDcopy:STARt (see page 202)	Starts a print job.
:HARDcopy:APRinter (see page 196)	Sets the active printer.
:HARDcopy:AREA (see page 195)	Specifies the area of the display to print (currently SCReen only).
:HARDcopy:INKSaver (see page 199)	Inverts screen colors to save ink when printing.
:HARDcopy:PRinter:LIST (see page 201)	Returns a list of the available printers.
:RECall Commands (see page 257)	Commands for recalling previously saved oscilloscope setups and traces.
:SAVE Commands (see page 262)	Commands for saving oscilloscope setups and traces, screen images, and data.
:SBUS Commands (see page 276)	Commands for controlling oscilloscope functions associated with the serial decode bus.
:TRIGger:CAN Commands (see page 324)	Commands for triggering on Controller Area Network (CAN) version 2.0A and 2.0B signals.
:TRIGger:IIC Commands (see page 355)	Commands for triggering on Inter-IC (IIC) signals.
:TRIGger:LIN Commands (see page 364)	Commands for triggering on Local Interconnect Network (LIN) signals.
:TRIGger:SPI Commands (see page 372)	Commands for triggering on Serial Peripheral Interface (SPI) signals.
:TRIGger:UART Commands (see page 387)	Commands for triggering on UART/RS-232 signals.
:WAVeform:SOURce:SUBSource (see page 424)	Selects subsource when :WAVeform:SOURce is SBUS (serial decode).

Changed Commands

Command	Differences
:TRIGger:MODE (see page 319)	You can now select the serial triggering modes.

Obsolete Commands

Obsolete Command	Current Command Equivalent	Behavior Differences
:HARDcopy:FILename (see page 473)	:RECall:FILename (see page 258) :SAVE:FILename (see page 258)	
:HARDcopy:FORMat (see page 474)	:HARDcopy:APRinter (see page 196) :SAVE:IMAGe:FORMat (see page 268) :SAVE:WAVeform:FORMat (see page 274)	
:HARDcopy:IGColors (see page 476)	:HARDcopy:INKSaver (see page 199)	
:HARDcopy:PDRiver (see page 477)	:HARDcopy:APRinter (see page 196)	

What's New in Version 4.10

New features in version 4.10 of the InfiniiVision 5000 Series oscilloscope software are:

- The square root waveform math function.
- Several new hardcopy printer drivers.

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

Changed Commands

Command	Differences
:FUNCtion:OPERation (see page 186)	You can now select the SQRT (square root) waveform math function.
:HARDcopy:PDRiver (see page 477)	You can now select the new DJPR0kx50, DJ55xx, PS470, and LJFastraster printer drivers.

Version 4.00 at Introduction

The Agilent InfiniiVision 5000 Series oscilloscopes were introduced with version 4.00 of oscilloscope operating software. The command set is similar to the 6000 Series oscilloscopes (and the 54620/54640 Series oscilloscopes before them) except that digital channels, rear-panel 10 Mhz reference BNC input/output, and serial bus triggering/decode features are not present.

2
Commands Quick Reference

Command Summary 24
Syntax Elements 56

Command Summary

Table 2 Common (*) Commands Summary

Command	Query	Options and Query Returns
`*CLS` (see page 65)	n/a	n/a

`*ESE <mask>` (see page 66)	`*ESE?` (see page 67)	`<mask> ::= 0 to 255; an integer in NR1 format:`
		
		`Bit Weight Name Enables`
		`--- ------ ---- ----------`
		` 7    128   PON  Power On`
		` 6     64   URQ  User Request`
		` 5     32   CME  Command Error`
		` 4     16   EXE  Execution Error`
		` 3      8   DDE  Dev. Dependent Error`
		` 2      4   QYE  Query Error`
		` 1      2   RQL  Request Control`
		` 0      1   OPC  Operation Complete`

n/a	`*ESR?` (see page 68)	`<status> ::= 0 to 255; an integer in NR1 format`

n/a	`*IDN?` (see page 68)	`AGILENT TECHNOLOGIES,<model>, <serial number>,X.XX.XX`
		`<model> ::= the model number of the instrument`
		`<serial number> ::= the serial number of the instrument`
		`<X.XX.XX> ::= the software revision of the instrument`

n/a	`*LRN?` (see page 71)	`<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format`
`*OPC` (see page 72)	`*OPC?` (see page 72)	`ASCII "1" is placed in the output queue when all pending device operations have completed.`

Table 2 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns

n/a	*OPT? (see page 73)	<return_value> ::= 0,0,<license info>
		<license info> ::= <All field>, <reserved>, <reserved>, <reserved>, <reserved>, <reserved>, <Low Speed Serial>, <Automotive Serial>, <reserved>, <Secure>, <reserved>, <reserved>, <reserved>, <reserved>, <RS-232/UART Serial>, <reserved>
		<All field> ::= {0 | All}
		<reserved> ::= 0
		<Low Speed Serial> ::= {0 | LSS}
		<Automotive Serial> ::= {0 | AMS}
		<Secure> ::= {0 | SEC}
		<RS-232/UART Serial> ::= {0 | 232}

*RCL <value> (see page 74)	n/a	<value> ::= {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}
*RST (see page 75)	n/a	See *RST (Reset) (see page 75)
*SAV <value> (see page 78)	n/a	<value> ::= {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

*SRE <mask> (see page 79)	*SRE? (see page 80)	<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)

Command	Query	Options and Query Returns
n/a	*STB? (see page 81)	<value> ::= 0 to 255; an integer in NR1 format, as shown in the following: Bit Weight Name "1" Indicates --- ------ ---- --------------- 7 128 OPER Operation status condition occurred. 6 64 RQS/MSS Instrument is requesting service. 5 32 ESB Enabled event status condition occurred. 4 16 MAV Message available. 3 8 ---- (Not used.) 2 4 MSG Message displayed. 1 2 USR User event condition occurred. 0 1 TRG A trigger occurred.
*TRG (see page 83)	n/a	n/a
n/a	*TST? (see page 84)	<result> ::= 0 or non-zero value; an integer in NR1 format
*WAI (see page 85)	n/a	n/a

Table 3 Root (:) Commands Summary

Command	Query	Options and Query Returns
n/a	:AER? (see page 89)	{0 | 1}; an integer in NR1 format
:AUToscale [<source>[,..,<source>]] (see page 90)	n/a	<source> ::= CHANnel<n> <source> can be repeated up to 5 times <n> ::= 1-2 or 1-4 in NR1 format
:AUToscale:AMODE <value> (see page 92)	:AUToscale:AMODE? (see page 92)	<value> ::= {NORMal | CURRent}}
:AUToscale:CHANnels <value> (see page 93)	:AUToscale:CHANnels? (see page 93)	<value> ::= {ALL | DISPlayed}}
:BLANk [<source>] (see page 94)	n/a	<source> ::= {CHANnel<n>} | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:CDISplay (see page 95)	n/a	n/a

Table 3 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:DIGitize [<source>[,..,<source>]] (see page 96)	n/a	<source> ::= {CHANnel<n> | FUNCtion | MATH} <source> can be repeated up to 5 times <n> ::= 1-2 or 1-4 in NR1 format
:HWEenable <n> (see page 98)	:HWEenable? (see page 98)	<n> ::= 16-bit integer in NR1 format
n/a	:HWERregister:CONDition? (see page 100)	<n> ::= 16-bit integer in NR1 format
n/a	:HWERegister[:EVENt]? (see page 102)	<n> ::= 16-bit integer in NR1 format
:MERGe <pixel memory> (see page 104)	n/a	<pixel memory> ::= {PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}}
:OPEE <n> (see page 105)	:OPEE? (see page 106)	<n> ::= 16-bit integer in NR1 format
n/a	:OPERregister:CONDition? (see page 107)	<n> ::= 16-bit integer in NR1 format
n/a	:OPERegister[:EVENt]? (see page 109)	<n> ::= 16-bit integer in NR1 format
:OVLenable <mask> (see page 111)	:OVLenable? (see page 112)	<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	:OVLRegister? (see page 113)	<value> ::= integer in NR1 format. See OVLenable for <value>
:PRINt [<options>] (see page 115)	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.

Table 3 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:RUN (see page 116)	n/a	n/a
n/a	:SERial (see page 117)	<return value> ::= unquoted string containing serial number
:SINGle (see page 118)	n/a	n/a

n/a	:STATus? <display> (see page 119)	{0 | 1}
		<display> ::= {CHANnel<n> | FUNCtion | MATH}
		<n> ::= 1-2 or 1-4 in NR1 format

:STOP (see page 120)	n/a	n/a
n/a	:TER? (see page 121)	{0 | 1}
:VIEW <source> (see page 122)	n/a	<source> ::= {CHANnel<n> | PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format

Table 4 :ACQuire Commands Summary

Command	Query	Options and Query Returns
n/a	:ACQuire:AALias? (see page 125)	{1 | 0}
:ACQuire:COMPlete <complete> (see page 126)	:ACQuire:COMPlete? (see page 126)	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNt <count> (see page 127)	:ACQuire:COUNt? (see page 127)	<count> ::= an integer from 1 to 65536 in NR1 format
:ACQuire:DAALias <mode> (see page 128)	:ACQuire:DAALias? (see page 128)	<mode> ::= {DISable | AUTO}
:ACQuire:MODE <mode> (see page 129)	:ACQuire:MODE? (see page 129)	<mode> ::= {RTIMe | ETIMe}
n/a	:ACQuire:POINts? (see page 130)	<# points> ::= an integer in NR1 format
n/a	:ACQuire:SRATe? (see page 131)	<sample_rate> ::= sample rate (samples/s) in NR3 format
:ACQuire:TYPE <type> (see page 132)	:ACQuire:TYPE? (see page 132)	<type> ::= {NORMal | AVERage | HRESolution | PEAK}

Table 5 :CALibrate Commands Summary

Command	Query	Options and Query Returns
n/a	:CALibrate:DATE? (see page 135)	<return value> ::= <day>,<month>,<year>; all in NR1 format
:CALibrate:LABel <string> (see page 136)	:CALibrate:LABel? (see page 136)	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:STARt (see page 137)	n/a	n/a
n/a	:CALibrate:STATus? (see page 138)	<return value> ::= ALL,<status_code>,<status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:SWITch? (see page 139)	{PROTected | UNPRotected}
n/a	:CALibrate:TEMPerature? (see page 140)	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see page 141)	<return value> ::= <hours>,<minutes>,<seconds>; all in NR1 format

Table 6 :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BWLimit {{0 | OFF} | {1 | ON}} (see page 145)	:CHANnel<n>:BWLimit? (see page 145)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:COUPling <coupling> (see page 146)	:CHANnel<n>:COUPling? (see page 146)	<coupling> ::= {AC | DC} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:DISPlay {{0 | OFF} | {1 | ON}} (see page 147)	:CHANnel<n>:DISPlay? (see page 147)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:IMPedance <impedance> (see page 148)	:CHANnel<n>:IMPedance? (see page 148)	<impedance> ::= {ONEMeg | FIFTy} <n> ::= 1-2 or 1-4 in NR1 format

Table 6 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:INVert {{0 | OFF} | {1 | ON}} (see page 149)	:CHANnel<n>:INVert? (see page 149)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:LABel <string> (see page 150)	:CHANnel<n>:LABel? (see page 150)	<string> ::= any series of 6 or less ASCII characters enclosed in quotation marks <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:OFFSet <offset>[suffix] (see page 151)	:CHANnel<n>:OFFSet? (see page 151)	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe <attenuation> (see page 152)	:CHANnel<n>:PROBe? (see page 152)	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4r in NR1 format
n/a	:CHANnel<n>:PROBe:ID? (see page 153)	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:PROBe:SKEW <skew_value> (see page 154)	:CHANnel<n>:PROBe:SKEW? (see page 154)	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:PROBe:STYPe <signal type> (see page 155)	:CHANnel<n>:PROBe:STYPe? (see page 155)	<signal type> ::= {DIFFerential | SINGle} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:PROTection (see page 156)	:CHANnel<n>:PROTection? (see page 156)	{NORM | TRIP} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:RANGe <range>[suffix] (see page 157)	:CHANnel<n>:RANGe? (see page 157)	<range> ::= Vertical full-scale range value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:SCALe <scale>[suffix] (see page 158)	:CHANnel<n>:SCALe? (see page 158)	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:UNITs <units> (see page 159)	:CHANnel<n>:UNITs? (see page 159)	<units> ::= {VOLT | AMPere} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:VERNier {{0 | OFF} | {1 | ON}} (see page 160)	:CHANnel<n>:VERNier? (see page 160)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format

Table 7 :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:CLEar (see page 163)	n/a	n/a
:DISPlay:DATA [<format>][,][<area>][,][<palette>]<display data> (see page 164)	:DISPlay:DATA? [<format>][,][<area>][,][<palette>] (see page 164)	<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 166)	:DISPlay:LABel? (see page 166)	{0 | 1}
:DISPlay:LABList <binary block> (see page 167)	:DISPlay:LABList? (see page 167)	<binary block> ::= an ordered list of up to 75 labels, each 6 characters maximum, separated by newline characters
:DISPlay:PERSistence <value> (see page 168)	:DISPlay:PERSistence? (see page 168)	<value> ::= {MINimum | INFinite}}
:DISPlay:SOURce <value> (see page 169)	:DISPlay:SOURce? (see page 169)	<value> ::= {PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}}
:DISPlay:VECTors {{1 | ON} | {0 | OFF}} (see page 170)	:DISPlay:VECTors? (see page 170)	{1 | 0}

Table 8 :EXTernal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXTernal:BWLimit <bwlimit> (see page 173)	:EXTernal:BWLimit? (see page 173)	<bwlimit> ::= {0 | OFF}
:EXTernal:IMPedance <value> (see page 174)	:EXTernal:IMPedance? (see page 174)	<impedance> ::= {ONEMeg | FIFTy}

Table 8 :EXTernal Trigger Commands Summary (continued)

Command	Query	Options and Query Returns
:EXTernal:PROBe <attenuation> (see page 175)	:EXTernal:PROBe? (see page 175)	<attenuation> ::= probe attenuation ratio in NR3 format
n/a	:EXTernal:PROBe:ID? (see page 176)	<probe id> ::= unquoted ASCII string up to 11 characters
:EXTernal:PROBe:STYPe <signal type> (see page 177)	:EXTernal:PROBe:STYPe? (see page 177)	<signal type> ::= {DIFFerential | SINGle}
:EXTernal:PROTection[:CLEar] (see page 178)	:EXTernal:PROTection? (see page 178)	{NORM | TRIP}
:EXTernal:RANGe <range>[<suffix>] (see page 179)	:EXTernal:RANGe? (see page 179)	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V | mV}
:EXTernal:UNITs <units> (see page 180)	:EXTernal:UNITs? (see page 180)	<units> ::= {VOLT | AMPere}

Table 9 :FUNCtion Commands Summary

Command	Query	Options and Query Returns
:FUNCtion:CENTer <frequency> (see page 183)	:FUNCtion:CENTer? (see page 183)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 25 GHz.
:FUNCtion:DISPlay {{0 | OFF} | {1 | ON}} (see page 184)	:FUNCtion:DISPlay? (see page 184)	{0 | 1}
:FUNCtion:OFFSet <offset> (see page 185)	:FUNCtion:OFFSet? (see page 185)	<offset> ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function.
:FUNCtion:OPERation <operation> (see page 186)	:FUNCtion:OPERation? (see page 186)	<operation> ::= {SUBTract | MULTiply | INTegrate | DIFFerentiate | FFT | SQRT}

Table 9 :FUNCtion Commands Summary (continued)

Command	Query	Options and Query Returns
:FUNCtion:RANGe <range> (see page 187)	:FUNCtion:RANGe? (see page 187)	<range> ::= the full-scale vertical axis value in NR3 format. The range for ADD, SUBT, MULT is 8E-6 to 800E+3. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFFerentiate function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV.
:FUNCtion:REFerence <level> (see page 188)	:FUNCtion:REFerence? (see page 188)	<level> ::= 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).
:FUNCtion:SCALe <scale value>[<suffix>] (see page 189)	:FUNCtion:SCALe? (see page 189)	<scale value> ::= integer in NR1 format <suffix> ::= {V | dB}
:FUNCtion:SOURce <source> (see page 190)	:FUNCtion:SOURce? (see page 190)	<source> ::= {CHANnel<n> | ADD | SUBT | MULT} <n> ::= 1-2 or 1-4 in NR1 format
:FUNCtion:SPAN <span> (see page 191)	:FUNCtion:SPAN? (see page 191)	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz.
:FUNCtion:WINDow <window> (see page 192)	:FUNCtion:WINDow? (see page 192)	<window> ::= {RECTangular | HANNing | FLATtop}

Table 10 :HARDcopy Commands Summary

Command	Query	Options and Query Returns
:HARDcopy:AREA <area> (see page 195)	:HARDcopy:AREA? (see page 195)	<area> ::= SCReen
:HARDcopy:APRinter <active_printer> (see page 196)	:HARDcopy:APRinter? (see page 196)	<active_printer> ::= {<index> | <name>} <index> ::= integer index of printer in list <name> ::= name of printer in list
:HARDcopy:FACTors {{0 | OFF} | {1 | ON}} (see page 197)	:HARDcopy:FACTors? (see page 197)	{0 | 1}
:HARDcopy:FFEed {{0 | OFF} | {1 | ON}} (see page 198)	:HARDcopy:FFEed? (see page 198)	{0 | 1}
:HARDcopy:INKSaver {{0 | OFF} | {1 | ON}} (see page 199)	:HARDcopy:INKSaver? (see page 199)	{0 | 1}
:HARDcopy:PALette <palette> (see page 200)	:HARDcopy:PALette? (see page 200)	<palette> ::= {COLor | GRAYscale | NONE}
n/a	:HARDcopy:PRinter:LIST? (see page 201)	<list> ::= [<printer_spec>] ... [printer_spec>] <printer_spec> ::= "<index>,<active>,<name>;" <index> ::= integer index of printer <active> ::= {Y | N} <name> ::= name of printer
:HARDcopy:STARt (see page 202)	n/a	n/a

Table 11 :MARKer Commands Summary

Command	Query	Options and Query Returns
:MARKer:MODE <mode> (see page 205)	:MARKer:MODE? (see page 205)	<mode> ::= {OFF | MEASurement | MANual}
:MARKer:X1Position <position>[suffix] (see page 206)	:MARKer:X1Position? (see page 206)	<position> ::= X1 cursor position value in NR3 format [suffix] ::= {s | ms | us | ns | ps | Hz | kHz | MHz} <return_value> ::= X1 cursor position value in NR3 format
:MARKer:X1Y1source <source> (see page 207)	:MARKer:X1Y1source? (see page 207)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>
:MARKer:X2Position <position>[suffix] (see page 208)	:MARKer:X2Position? (see page 208)	<position> ::= X2 cursor position value in NR3 format [suffix] ::= {s | ms | us | ns | ps | Hz | kHz | MHz} <return_value> ::= X2 cursor position value in NR3 format
:MARKer:X2Y2source <source> (see page 209)	:MARKer:X2Y2source? (see page 209)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>
n/a	:MARKer:XDELta? (see page 210)	<return_value> ::= X cursors delta value in NR3 format
:MARKer:Y1Position <position>[suffix] (see page 211)	:MARKer:Y1Position? (see page 211)	<position> ::= Y1 cursor position value in NR3 format [suffix] ::= {V | mV | dB} <return_value> ::= Y1 cursor position value in NR3 format
:MARKer:Y2Position <position>[suffix] (see page 212)	:MARKer:Y2Position? (see page 212)	<position> ::= Y2 cursor position value in NR3 format [suffix] ::= {V | mV | dB} <return_value> ::= Y2 cursor position value in NR3 format
n/a	:MARKer:YDELta? (see page 213)	<return_value> ::= Y cursors delta value in NR3 format

Table 12 :MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:CLEar (see page 220)	n/a	n/a
:MEASure:COUNter [<source>] (see page 221)	:MEASure:COUNter? [<source>] (see page 221)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= counter frequency in Hertz in NR3 format
:MEASure:DEFine DELay, <delay spec> (see page 222)	:MEASure:DEFine? DELay (see page 223)	<delay spec> ::= <edge_spec1>,<edge_spec2> edge_spec1 ::= [<slope>]<occurrence> edge_spec2 ::= [<slope>]<occurrence> <slope> ::= {+ | -} <occurrence> ::= integer

:MEASure:DEFine THResholds, <threshold spec> (see page 222)	:MEASure:DEFine? THResholds (see page 223)	<threshold spec> ::= {STANdard} | {<threshold mode>,<upper>, <middle>,<lower>}
		<threshold mode> ::= {PERCent | ABSolute}

:MEASure:DELay [<source1>] [,<source2>] (see page 225)	:MEASure:DELay? [<source1>] [,<source2>] (see page 225)	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUTYcycle [<source>] (see page 227)	:MEASure:DUTYcycle? [<source>] (see page 227)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format
:MEASure:FALLtime [<source>] (see page 228)	:MEASure:FALLtime? [<source>] (see page 228)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds between the lower and upper thresholds in NR3 format
:MEASure:FREQuency [<source>] (see page 229)	:MEASure:FREQuency? [<source>] (see page 229)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= frequency in Hertz in NR3 format

Table 12 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:NWIDth [<source>] (see page 230)	:MEASure:NWIDth? [<source>] (see page 230)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>] (see page 231)	:MEASure:OVERshoot? [<source>] (see page 231)	<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
:MEASure:PERiod [<source>] (see page 233)	:MEASure:PERiod? [<source>] (see page 233)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source1>] [,<source2>] (see page 234)	:MEASure:PHASe? [<source1>] [,<source2>] (see page 234)	<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
:MEASure:PREShoot [<source>] (see page 235)	:MEASure:PREShoot? [<source>] (see page 235)	<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
:MEASure:PWIDth [<source>] (see page 236)	:MEASure:PWIDth? [<source>] (see page 236)	<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
:MEASure:RISEtime [<source>] (see page 237)	:MEASure:RISEtime? [<source>] (see page 237)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= rise time in seconds in NR3 format

Table 12 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:SDEViation [<source>] (see page 238)	:MEASure:SDEViation? [<source>] (see page 238)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated std deviation in NR3 format
:MEASure:SHOW {1 | ON} (see page 239)	:MEASure:SHOW? (see page 239)	{1}

:MEASure:SOURce [<source1>] [,<source2>] (see page 240)	:MEASure:SOURce? (see page 240)	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH}
		<n> ::= 1-2 or 1-4 in NR1 format
		<return_value> ::= {<source> | NONE}

n/a	:MEASure:TEDGe? <slope><occurrence>[, <source>] (see page 242)	<slope> ::= direction of the waveform <occurrence> ::= the transition to be reported <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds of the specified transition
n/a	:MEASure:TVALue? <value>, [<slope>]<occurrence> [,<source>] (see page 244)	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <return_value> ::= time in seconds of specified voltage crossing in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:MEASure:VAMPlitude [<source>] (see page 246)	:MEASure:VAMPlitude? [<source>] (see page 246)	<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

Table 12 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAVerage [<source>] (see page 247)	:MEASure:VAVerage? [<source>] (see page 247)	<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 248)	:MEASure:VBASe? [<source>] (see page 248)	<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 249)	:MEASure:VMAX? [<source>] (see page 249)	<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 250)	:MEASure:VMIN? [<source>] (see page 250)	<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 251)	:MEASure:VPP? [<source>] (see page 251)	<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
:MEASure:VRMS [<source>] (see page 252)	:MEASure:VRMS? [<source>] (see page 252)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated dc RMS voltage in NR3 format
n/a	:MEASure:VTIMe? <vtime>[,<source>] (see page 253)	<vtime> ::= displayed time from trigger in seconds in NR3 format <return_value> ::= voltage at the specified time in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format

Table 12 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VTOP [<source>] (see page 254)	:MEASure:VTOP? [<source>] (see page 254)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= voltage at the top of the waveform in NR3 format
:MEASure:XMAX [<source>] (see page 255)	:MEASure:XMAX? [<source>] (see page 255)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format
:MEASure:XMIN [<source>] (see page 256)	:MEASure:XMIN? [<source>] (see page 256)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format

Table 13 :RECall Commands Summary

Command	Query	Options and Query Returns
:RECall:FILename <base_name> (see page 258)	:RECall:FILename? (see page 258)	<base_name> ::= quoted ASCII string
:RECall:IMAGe[:STARt] [<file_spec>] (see page 259)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
n/a	:RECall:PWD? (see page 260)	<path_info> ::= quoted ASCII string
:RECall:SETup[:STARt] [<file_spec>] (see page 261)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string

Table 14 :SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:FILename <base_name> (see page 264)	:SAVE:FILename? (see page 264)	<base_name> ::= quoted ASCII string
:SAVE:IMAGe[:STARt] [<file_spec>] (see page 265)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:IMAGe:AREA <area> (see page 266)	:SAVE:IMAGe:AREA? (see page 266)	<area> ::= {GRATicule | SCReen}
:SAVE:IMAGe:FACTors {{0 | OFF} | {1 | ON}} (see page 267)	:SAVE:IMAGe:FACTors? (see page 267)	{0 | 1}
:SAVE:IMAGe:FORMat <format> (see page 268)	:SAVE:IMAGe:FORMat? (see page 268)	<format> ::= {TIFF | {BMP | BMP24bit} | BMP8bit | PNG | NONE}
:SAVE:IMAGe:INKSaver {{0 | OFF} | {1 | ON}} (see page 269)	:SAVE:IMAGe:INKSaver? (see page 269)	{0 | 1}
:SAVE:IMAGe:PALette <palette> (see page 270)	:SAVE:IMAGe:PALette? (see page 270)	<palette> ::= {COLor | GRAYscale | MONochrome}
n/a	:SAVE:PWD? (see page 271)	<path_info> ::= quoted ASCII string
:SAVE:SETup[:STARt] [<file_spec>] (see page 272)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:WAVeform[:STARt] [<file_name>] (see page 273)	n/a	<file_name> ::= quoted ASCII string
:SAVE:WAVeform:FORMat <format> (see page 274)	:SAVE:WAVeform:FORMat? (see page 274)	<format> ::= {ALB | ASCiixy | CSV | BINary | NONE}
:SAVE:WAVeform:LENGth <length> (see page 275)	:SAVE:WAVeform:LENGth? (see page 275)	<length> ::= 100 to max. length; an integer in NR1 format

Table 15 :SBUS Commands Summary

Command	Query	Options and Query Returns
n/a	:SBUS:CAN:COUNt:ERRor? (see page 278)	<frame_count> ::= integer in NR1 format
n/a	:SBUS:CAN:COUNt:OVERload? (see page 279)	<frame_count> ::= integer in NR1 format
:SBUS:CAN:COUNt:RESet (see page 280)	n/a	n/a
n/a	:SBUS:CAN:COUNt:TOTal? (see page 281)	<frame_count> ::= integer in NR1 format
n/a	:SBUS:CAN:COUNt:UTILization? (see page 282)	<percent> ::= floating-point in NR3 format
:SBUS:DISPlay {{0 | OFF} | {1 | ON}} (see page 283)	:SBUS:DISPlay? (see page 283)	{0 | 1}
:SBUS:IIC:ASIZe <size> (see page 284)	:SBUS:IIC:ASIZe? (see page 284)	<size> ::= {BIT7 | BIT8}
:SBUS:LIN:PARity {{0 | OFF} | {1 | ON}} (see page 285)	:SBUS:LIN:PARity? (see page 285)	{0 | 1}
:SBUS:MODE <mode> (see page 286)	:SBUS:MODE? (see page 286)	<mode> ::= {IIC | SPI | CAN | LIN | FLEXray | UART}
:SBUS:SPI:WIDTh <word_width> (see page 287)	:SBUS:SPI:WIDTh? (see page 287)	<word_width> ::= integer 4-16 in NR1 format
:SBUS:UART:BASE <base> (see page 288)	:SBUS:UART:BASE? (see page 288)	<base> ::= {ASCii | BINary | HEX}
n/a	:SBUS:UART:COUNt:ERRor? (see page 289)	<frame_count> ::= integer in NR1 format
:SBUS:UART:COUNt:RESet (see page 290)	n/a	n/a
n/a	:SBUS:UART:COUNt:RXFRames? (see page 291)	<frame_count> ::= integer in NR1 format

Table 15 :SBUS Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:SBUS:UART:COUNt:TXFRames? (see page 292)	<frame_count> ::= integer in NR1 format
:SBUS:UART:FRAMing <value> (see page 293)	:SBUS:UART:FRAMing? (see page 293)	<value> ::= {OFF | <decimal> | <nondecimal>} <decimal> ::= 8-bit integer from 0-255 (0x00-0xff) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

Table 16 :SYSTem Commands Summary

Command	Query	Options and Query Returns

:SYSTem:DATE <date> (see page 295)	:SYSTem:DATE? (see page 295)	<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 296)	n/a	<string> ::= up to 254 characters as a quoted ASCII string
n/a	:SYSTem:ERRor? (see page 297)	<error> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see page 499).
:SYSTem:LOCK (see page 298)	:SYSTem:LOCK? (see page 298)	<value> ::= {ON | OFF}
:SYSTem:SETup <setup_data> (see page 299)	:SYSTem:SETup? (see page 299)	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:TIME <time> (see page 301)	:SYSTem:TIME? (see page 301)	<time> ::= hours,minutes,seconds in NR1 format

Table 17 :TIMebase Commands Summary

Command	Query	Options and Query Returns
:TIMebase:MODE <value> (see page 304)	:TIMebase:MODE? (see page 304)	<value> ::= {MAIN | WINDow | XY | ROLL}
:TIMebase:POSition <pos> (see page 305)	:TIMebase:POSition? (see page 305)	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMebase:RANGe <range_value> (see page 306)	:TIMebase:RANGe? (see page 306)	<range_value> ::= 10 ns through 500 s in NR3 format
:TIMebase:REFerence {LEFT | CENTer | RIGHt} (see page 307)	:TIMebase:REFerence? (see page 307)	<return_value> ::= {LEFT | CENTer | RIGHt}
:TIMebase:SCALe <scale_value> (see page 308)	:TIMebase:SCALe? (see page 308)	<scale_value> ::= scale value in seconds in NR3 format
:TIMebase:VERNier {{0 | OFF} | {1 | ON}} (see page 309)	:TIMebase:VERNier? (see page 309)	{0 | 1}
:TIMebase:WINDow:POSition <pos> (see page 310)	:TIMebase:WINDow:POSition? (see page 310)	<pos> ::= time from the trigger event to the delayed view reference point in NR3 format
:TIMebase:WINDow:RANGe <range_value> (see page 311)	:TIMebase:WINDow:RANGe? (see page 311)	<range value> ::= range value in seconds in NR3 format for the delayed window
:TIMebase:WINDow:SCALe <scale_value> (see page 312)	:TIMebase:WINDow:SCALe? (see page 312)	<scale_value> ::= scale value in seconds in NR3 format for the delayed window

Table 18 General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:HFReject {{0 | OFF} | {1 | ON}} (see page 317)	:TRIGger:HFReject? (see page 317)	{0 | 1}
:TRIGger:HOLDoff <holdoff_time> (see page 318)	:TRIGger:HOLDoff? (see page 318)	<holdoff_time> ::= 60 ns to 10 s in NR3 format

Table 18 General :TRIGger Commands Summary (continued)

Command	Query	Options and Query Returns

:TRIGger:MODE <mode> (see page 319)	:TRIGger:MODE? (see page 319)	<mode> ::= {EDGE | GLITch | PATTern | DURation | TV}
		<return_value> ::= {<mode> | <none>}
		<none> ::= query returns "NONE" if the :TIMebase:MODE is ROLL or XY

:TRIGger:NREJect {{0 | OFF} | {1 | ON}} (see page 320)	:TRIGger:NREJect? (see page 320)	{0 | 1}
:TRIGger:PATTern <value>, <mask> [,<edge source>,<edge>] (see page 321)	:TRIGger:PATTern? (see page 321)	<value> ::= integer in NR1 format or <string> <mask> ::= integer in NR1 format or <string> <string> ::= "0xnn"; n ::= {0,..,9 | A,..,F} (# bits = # channels) <edge source> ::= {CHANnel<n> | EXTernal | NONE} <edge> ::= {POSitive | NEGative} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SWEep <sweep> (see page 323)	:TRIGger:SWEep? (see page 323)	<sweep> ::= {AUTO | NORMal}

Table 19 :TRIGger:CAN Commands Summary

Command	Query	Options and Query Returns
:TRIGger:CAN:PATTern:DATA <value>, <mask> (see page 326)	:TRIGger:CAN:PATTern:DATA? (see page 326)	<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 327)	:TRIGger:CAN:PATTern:DATA:LENGth? (see page 327)	<length> ::= integer from 1 to 8 in NR1 format (with Option AMS)
:TRIGger:CAN:PATTern:ID <value>, <mask> (see page 328)	:TRIGger:CAN:PATTern:ID? (see page 328)	<value> ::= 32-bit integer in decimal, <nondecimal>, or <string> (with Option AMS) <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
:TRIGger:CAN:PATTern:ID:MODE <value> (see page 329)	:TRIGger:CAN:PATTern:ID:MODE? (see page 329)	<value> ::= {STANdard | EXTended} (with Option AMS)
:TRIGger:CAN:SAMPlepoint <value> (see page 330)	:TRIGger:CAN:SAMPlepoint? (see page 330)	<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
:TRIGger:CAN:SIGNal:BAUDrate <baudrate> (see page 331)	:TRIGger:CAN:SIGNal:BAUDrate? (see page 331)	<baudrate> ::= {10000 | 20000 | 33300 | 50000 | 62500 | 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000}

Table 19 :TRIGger:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:CAN:SOURce <source> (see page 332)	:TRIGger:CAN:SOURce? (see page 332)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 |} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:CAN:TRIGger <condition> (see page 333)	:TRIGger:CAN:TRIGger? (see page 334)	<condition> ::= {SOF} (without Option AMS) <condition> ::= {SOF | DATA | ERRor | IDData | IDEither | IDRemote | ALLerrors | OVERload | ACKerror} (with Option AMS)

Table 20 :TRIGger:DURation Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DURation:GREaterthan <greater than time>[suffix] (see page 336)	:TRIGger:DURation:GREaterthan? (see page 336)	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:DURation:LESSthan <less than time>[suffix] (see page 337)	:TRIGger:DURation:LESSthan? (see page 337)	<less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:DURation:PATTern <value>, <mask> (see page 338)	:TRIGger:DURation:PATTern? (see page 338)	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= ""0xnnnnnn"" n ::= {0,..,9 | A,..,F}
:TRIGger:DURation:QUALifier <qualifier> (see page 339)	:TRIGger:DURation:QUALifier? (see page 339)	<qualifier> ::= {GREaterthan | LESSthan | INRange | OUTRange | TIMeout}
:TRIGger:DURation:RANGe <greater than time>[suffix], <less than time>[suffix] (see page 340)	:TRIGger:DURation:RANGe? (see page 340)	<greater than time> ::= min duration from 10 ns to 9.99 seconds in NR3 format <less than time> ::= max duration from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}

Table 21 :TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:COUPling {AC | DC | LF} (see page 342)	:TRIGger[:EDGE]:COUPling? (see page 342)	{AC | DC | LF}
:TRIGger[:EDGE]:LEVel <level> [,<source>] (see page 343)	:TRIGger[:EDGE]:LEVel? [<source>] (see page 343)	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. <source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger[:EDGE]:REJect {OFF | LF | HF} (see page 344)	:TRIGger[:EDGE]:REJect? (see page 344)	{OFF | LF | HF}
:TRIGger[:EDGE]:SLOPe <polarity> (see page 345)	:TRIGger[:EDGE]:SLOPe? (see page 345)	<polarity> ::= {POSitive | NEGative | EITHer | ALTernate}
:TRIGger[:EDGE]:SOURce <source> (see page 346)	:TRIGger[:EDGE]:SOURce? (see page 346)	<source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format

Table 22 :TRIGger:GLITch Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITch:GREaterthan <greater than time>[suffix] (see page 348)	:TRIGger:GLITch:GREaterthan? (see page 348)	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:LESSthan <less than time>[suffix] (see page 349)	:TRIGger:GLITch:LESSthan? (see page 349)	<less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}

Table 22 :TRIGger:GLITch Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:GLITch:LEVel <level> [<source>] (see page 350)	:TRIGger:GLITch:LEVel? (see page 350)	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. <source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:GLITch:POLarity <polarity> (see page 351)	:TRIGger:GLITch:POLarity? (see page 351)	<polarity> ::= {POSitive | NEGative}
:TRIGger:GLITch:QUALifier <qualifier> (see page 352)	:TRIGger:GLITch:QUALifier? (see page 352)	<qualifier> ::= {GREaterthan | LESSthan | RANGe}
:TRIGger:GLITch:RANGe <greater than time>[suffix], <less than time>[suffix] (see page 353)	:TRIGger:GLITch:RANGe? (see page 353)	<greater than time> ::= start time from 10 ns to 9.99 seconds in NR3 format <less than time> ::= stop time from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:SOURce <source> (see page 354)	:TRIGger:GLITch:SOURce? (see page 354)	<source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format

Table 23 :TRIGger:IIC Commands Summary

Command	Query	Options and Query Returns
:TRIGger:IIC:PATTern:ADDRess <value> (see page 356)	:TRIGger:IIC:PATTern:ADDRess? (see page 356)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}
:TRIGger:IIC:PATTern:DATA <value> (see page 357)	:TRIGger:IIC:PATTern:DATA? (see page 357)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}
:TRIGger:IIC:PATTern:DATa2 <value> (see page 358)	:TRIGger:IIC:PATTern:DATa2? (see page 358)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}

Table 23 :TRIGger:IIC Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:IIC[:SOURce]:CLOCk <source> (see page 359)	:TRIGger:IIC[:SOURce]:CLOCk? (see page 359)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 } for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC[:SOURce]:DATA <source> (see page 360)	:TRIGger:IIC[:SOURce]:DATA? (see page 360)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 } for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC:TRIGger:QUALifier <value> (see page 361)	:TRIGger:IIC:TRIGger:QUALifier? (see page 361)	<value> ::= {EQUal | NOTequal | LESSthan | GREaterthan}
:TRIGger:IIC:TRIGger[:TYPE] <type> (see page 362)	:TRIGger:IIC:TRIGger[:TYPE]? (see page 362)	<type> ::= {STARt | STOP | READ7 | READEprom | WRITe7 | WRITe10 | NACKnowledge | ANACknowledge | R7Data2 | W7Data2 | RESTart}

Table 24 :TRIGger:LIN Commands Summary

Command	Query	Options and Query Returns
:TRIGger:LIN:ID <value> (see page 365)	:TRIGger:LIN:ID? (see page 365)	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f (with Option AMS) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary <string> ::= "0xnn" where n ::= {0,..,9 | A,..,F} for hexadecimal
:TRIGger:LIN:SAMPlepoint <value> (see page 366)	:TRIGger:LIN:SAMPlepoint? (see page 366)	<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
:TRIGger:LIN:SIGNal:BAUDrate <baudrate> (see page 367)	:TRIGger:LIN:SIGNal:BAUDrate? (see page 367)	<baudrate> ::= {2400 | 9600 | 19200}

Table 24 :TRIGger:LIN Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:LIN:SOURce <source> (see page 368)	:TRIGger:LIN:SOURce? (see page 368)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:LIN:STANdard <std> (see page 369)	:TRIGger:LIN:STANdard ? (see page 369)	<std> ::= {LIN13 | LIN20}
:TRIGger:LIN:SYNCbreak <value> (see page 370)	:TRIGger:LIN:SYNCbreak? (see page 370)	<value> ::= integer = {11 | 12 | 13}
:TRIGger:LIN:TRIGger <condition> (see page 371)	:TRIGger:LIN:TRIGger? (see page 371)	<condition> ::= {SYNCbreak} (without Option AMS) <condition> ::= {SYNCbreak | ID} (with Option AMS)

Table 25 :TRIGger:TV Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TV:LINE <line number> (see page 382)	:TRIGger:TV:LINE? (see page 382)	<line number> ::= integer in NR1 format
:TRIGger:TV:MODE <tv mode> (see page 383)	:TRIGger:TV:MODE? (see page 383)	<tv mode> ::= {FIEld1 | FIEld2 | AFIelds | ALINes | LINE | VERTical | LFIeld1 | LFIeld2 | LALTernate | LVERtical}
:TRIGger:TV:POLarity <polarity> (see page 384)	:TRIGger:TV:POLarity? (see page 384)	<polarity> ::= {POSitive | NEGative}
:TRIGger:TV:SOURce <source> (see page 385)	:TRIGger:TV:SOURce? (see page 385)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 integer in NR1 format
:TRIGger:TV:STANdard <standard> (see page 386)	:TRIGger:TV:STANdard? (see page 386)	<standard> ::= {GENeric | NTSC | PALM | PAL | SECam | {P480L60HZ | P480} | {P720L60HZ | P720} | {P1080L24HZ | P1080} | P1080L25HZ | {I1080L50HZ | I1080} | I1080L60HZ}

Table 26 :TRIGger:UART Commands Summary

Command	Query	Options and Query Returns
:TRIGger:UART:BAUDrate <baudrate> (see page 389)	:TRIGger:UART:BAUDrate? (see page 389)	<baudrate> ::= {1200 | 1800 | 2000 | 2400 | 3600 | 4800 | 7200 | 9600 | 14400 | 15200 | 19200 | 28800 | 38400 | 56000 | 57600 | 76800 | 115200 | 128000 | 230400 | 460800 | 921600 | 1382400 | 1843200 | 2764800}
:TRIGger:UART:BITorder <bitorder> (see page 390)	:TRIGger:UART:BITorder? (see page 390)	<bitorder> ::= {LSBFirst | MSBFirst}
:TRIGger:UART:BURSt <value> (see page 391)	:TRIGger:UART:BURSt? (see page 391)	<value> ::= {OFF | 1 to 4096 in NR1 format}
:TRIGger:UART:DATA <value> (see page 392)	:TRIGger:UART:DATA? (see page 392)	<value> ::= 8-bit integer in decimal or <nondecimal> from 0-255 (0x00-0xff) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
:TRIGger:UART:IDLE <time_value> (see page 393)	:TRIGger:UART:IDLE? (see page 393)	<time_value> ::= time from 10 us to 10 s in NR3 format
:TRIGger:UART:PARity <parity> (see page 394)	:TRIGger:UART:PARity? (see page 394)	<parity> ::= {EVEN | ODD | NONE}
:TRIGger:UART:POLarity <polarity> (see page 395)	:TRIGger:UART:POLarity? (see page 395)	<polarity> ::= {HIGH | LOW}
:TRIGger:UART:QUALifier <value> (see page 396)	:TRIGger:UART:QUALifier? (see page 396)	<value> ::= {EQUal | NOTequal | GREaterthan | LESSthan}
:TRIGger:UART:SOURce:RX <source> (see page 397)	:TRIGger:UART:SOURce:RX? (see page 397)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format

Table 26 :TRIGger:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:UART:SOURce:TX <source> (see page 398)	:TRIGger:UART:SOURce:TX? (see page 398)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:UART:TYPE <value> (see page 399)	:TRIGger:UART:TYPE? (see page 399)	<value> ::= {RSTArt | RSTop | RDATa | RD1 | RD0 | RDX | PARityerror | TSTArt | TSTOp | TDATa | TD1 | TD0 | TDX}
:TRIGger:UART:WIDTh <width> (see page 400)	:TRIGger:UART:WIDTh? (see page 400)	<width> ::= {5 | 6 | 7 | 8 | 9}

Table 27 :WAVeform Commands Summary

Command	Query	Options and Query Returns
:WAVeform:BYTeorder <value> (see page 408)	:WAVeform:BYTeorder? (see page 408)	<value> ::= {LSBFirst | MSBFirst}
n/a	:WAVeform:COUNt? (see page 409)	<count> ::= an integer from 1 to 65536 in NR1 format
n/a	:WAVeform:DATA? (see page 410)	<binary block length bytes>, <binary data> For example, to transmit 1000 bytes of data, the syntax would be: #800001000<1000 bytes of data><NL> 8 is the number of digits that follow 00001000 is the number of bytes to be transmitted <1000 bytes of data> is the actual data
:WAVeform:FORMat <value> (see page 412)	:WAVeform:FORMat? (see page 412)	<value> ::= {WORD | BYTE | ASCII}

Table 27 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVeform:POINts <# points> (see page 413)	:WAVeform:POINts? (see page 413)	<# points> ::= {100 | 250 | 500 | 1000 | <points_mode>} if waveform points mode is NORMal <# points> ::= {100 | 250 | 500 | 1000 | 2000 ... 8000000 in 1-2-5 sequence | <points_mode>} if waveform points mode is MAXimum or RAW <points_mode> ::= {NORMal | MAXimum | RAW}
:WAVeform:POINts:MODE <points_mode> (see page 415)	:WAVeform:POINts:MODE? (see page 415)	<points_mode> ::= {NORMal | MAXimum | RAW}
n/a	:WAVeform:PREamble? (see page 417)	<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 420)	:WAVeform:SOURce? (see page 420)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:WAVeform:SOURce:SUBSource <subsource> (see page 424)	:WAVeform:SOURce:SUBSource? (see page 424)	<subsource> ::= {{NONE | RX} | TX}
n/a	:WAVeform:TYPE? (see page 425)	<return_mode> ::= {NORM | PEAK | AVER | HRES}

Table 27 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
:WAVeform:UNSigned {{0 | OFF} | {1 | ON}} (see page 426)	:WAVeform:UNSigned? (see page 426)	{0 | 1}
:WAVeform:VIEW <view> (see page 427)	:WAVeform:VIEW? (see page 427)	<view> ::= {MAIN}
n/a	:WAVeform:XINCrement? (see page 428)	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVeform:XORigin? (see page 429)	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVeform:XREFerence? (see page 430)	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)
n/a	:WAVeform:YINCrement? (see page 431)	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVeform:YORigin? (see page 432)	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVeform:YREFerence? (see page 433)	<return_value> ::= y-reference value in the current preamble in NR1 format

Syntax Elements

- "Number Format" on page 56
- "<NL> (Line Terminator)" on page 56
- "[ ] (Optional Syntax Terms)" on page 56
- "{ } (Braces)" on page 56
- "::= (Defined As)" on page 56
- "< > (Angle Brackets)" on page 57
- "... (Ellipsis)" on page 57
- "n,..,p (Value Ranges)" on page 57
- "d (Digits)" on page 57
- "Quoted ASCII String" on page 57
- "Definite-Length Block Response Data" on page 57

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:

```
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

3
Commands by Subsystem

Subsystem	Description
"Common (*) Commands" on page 61	Commands defined by IEEE 488.2 standard that are common to all instruments.
"Root (:) Commands" on page 86	Control many of the basic functions of the oscilloscope and reside at the root level of the command tree.
":ACQuire Commands" on page 123	Set the parameters for acquiring and storing data.
":CALibrate Commands" on page 134	Utility commands for determining the state of the calibration factor protection switch.
":CHANnel<n> Commands" on page 142	Control all oscilloscope functions associated with individual analog channels or groups of channels.
":DISPlay Commands" on page 161	Control how waveforms, graticule, and text are displayed and written on the screen.
":EXTernal Trigger Commands" on page 171	Control the input characteristics of the external trigger input.
":FUNCtion Commands" on page 181	Control functions in the measurement/storage module.
":HARDcopy Commands" on page 193	Set and query the selection of hardcopy device and formatting options.
":MARKer Commands" on page 203	Set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors).
":MEASure Commands" on page 214	Select automatic measurements to be made and control time markers.
":RECall Commands" on page 257	Recall previously saved oscilloscope setups and traces.
":SAVE Commands" on page 262	Save oscilloscope setups and traces, screen images, and data.

Subsystem	Description
":SBUS Commands" on page 276	Control oscilloscope functions associated with the serial decode bus.
":SYSTem Commands" on page 294	Control basic system functions of the oscilloscope.
":TIMebase Commands" on page 302	Control all horizontal sweep functions.
":TRIGger Commands" on page 313	Control the trigger modes and parameters for each trigger type.
":WAVeform Commands" on page 401	Provide access to waveform data.

Command Types

Three types of commands are used:

- **Common (*) Commands** — See "Introduction to Common (*) Commands" on page 63 for more information.
- **Root Level (:) Commands** — See "Introduction to Root (:) Commands" on page 88 for more information.
- **Subsystem Commands** — Subsystem commands are grouped together under a common node of the "Command Tree" on page 535, 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 63.

Table 28 Common (*) Commands Summary

Command	Query	Options and Query Returns
`*CLS (see page 65)`	n/a	n/a

`*ESE <mask> (see page 66)`	`*ESE? (see page 67)`	`<mask> ::= 0 to 255; an integer in NR1 format:`
		
		`Bit Weight Name Enables`
		`--- ------ ---- ----------`
		`7 128 PON Power On`
		`6 64 URQ User Request`
		`5 32 CME Command Error`
		`4 16 EXE Execution Error`
		`3 8 DDE Dev. Dependent Error`
		`2 4 QYE Query Error`
		`1 2 RQL Request Control`
		`0 1 OPC Operation Complete`

n/a	`*ESR? (see page 68)`	`<status> ::= 0 to 255; an integer in NR1 format`

n/a	`*IDN? (see page 68)`	`AGILENT TECHNOLOGIES,<model>, <serial number>,X.XX.XX`
		`<model> ::= the model number of the instrument`
		`<serial number> ::= the serial number of the instrument`
		`<X.XX.XX> ::= the software revision of the instrument`

n/a	`*LRN? (see page 71)`	`<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format`
`*OPC (see page 72)`	`*OPC? (see page 72)`	`ASCII "1" is placed in the output queue when all pending device operations have completed.`

Table 28 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns

n/a	*OPT? (see page 73)	<return_value> ::= 0,0,<license info>
		<license info> ::= <All field>, <reserved>, <reserved>, <reserved>, <reserved>, <reserved>, <Low Speed Serial>, <Automotive Serial>, <reserved>, <Secure>, <reserved>, <reserved>, <reserved>, <reserved>, <RS-232/UART Serial>, <reserved>
		<All field> ::= {0 | All}
		<reserved> ::= 0
		<Low Speed Serial> ::= {0 | LSS}
		<Automotive Serial> ::= {0 | AMS}
		<Secure> ::= {0 | SEC}
		<RS-232/UART Serial> ::= {0 | 232}

*RCL <value> (see page 74)	n/a	<value> ::= {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}
*RST (see page 75)	n/a	See *RST (Reset) (see page 75)
*SAV <value> (see page 78)	n/a	<value> ::= {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

*SRE <mask> (see page 79)	*SRE? (see page 80)	<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 28 Common (*) Commands Summary (continued)

Command	Query	Options and Query Returns

n/a	*STB? (see page 81)	<value> ::= 0 to 255; an integer in NR1 format, as shown in the following:
		`Bit Weight Name "1" Indicates`
		`--- ------ ---- ---------------`
		` 7    128   OPER Operation status condition occurred.`
		` 6     64   RQS/MSS Instrument is requesting service.`
		` 5     32   ESB  Enabled event status condition occurred.`
		` 4     16   MAV  Message available.`
		` 3      8   ---- (Not used.)`
		` 2      4   MSG  Message displayed.`
		` 1      2   USR  User event condition occurred.`
		` 0      1   TRG  A trigger occurred.`

*TRG (see page 83)	n/a	n/a
n/a	*TST? (see page 84)	<result> ::= 0 or non-zero value; an integer in NR1 format
*WAI (see page 85)	n/a	n/a

Introduction to Common (*) Commands

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

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

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

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

*CLS (Clear Status)

C (see page 530)

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 63
- "*STB (Read Status Byte)" on page 81
- "*ESE (Standard Event Status Enable)" on page 66
- "*ESR (Standard Event Status Register)" on page 68
- "*SRE (Service Request Enable)" on page 79
- ":SYSTem:ERRor" on page 297

*ESE (Standard Event Status Enable)

C (see page 530)

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.

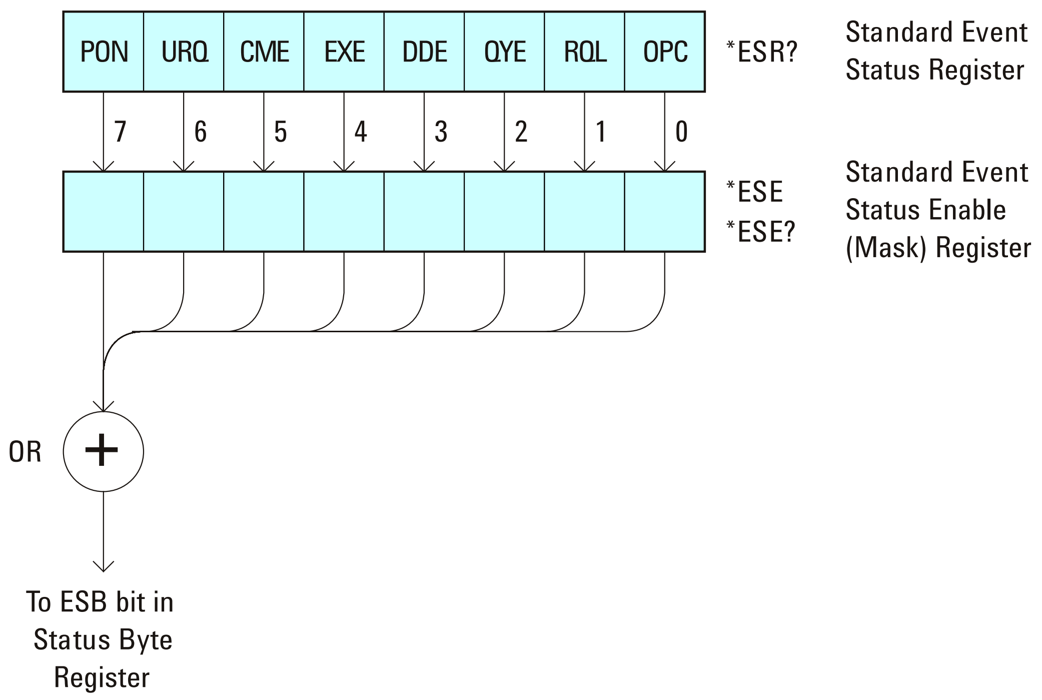

PON
URQ
CME
EXE
DDE
QYE
RQL
OPC
*ESR?
Standard Event Status Register
7
6
5
4
3
2
1
0
*ESE
*ESE?
Standard Event Status Enable (Mask) Register
OR
+
To ESB bit in Status Byte Register

Table 29 Standard Event Status Enable (ESE)

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

Table 29 Standard Event Status Enable (ESE) (continued)

Bit	Name	Description	When Set (1 = High = True), Enables:
1	RQL	Request Control	Event when the device is requesting control. (Not used.)
0	OPC	Operation Complete	Event when an operation is complete.

Query Syntax

```
*ESE?
```

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

Return Format

```
<mask_argument><NL>
```

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

See Also

- "Introduction to Common (*) Commands" on page 63
- "*ESR (Standard Event Status Register)" on page 68
- "*OPC (Operation Complete)" on page 72
- "*CLS (Clear Status)" on page 65

*ESR (Standard Event Status Register)

C (see page 530)

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.

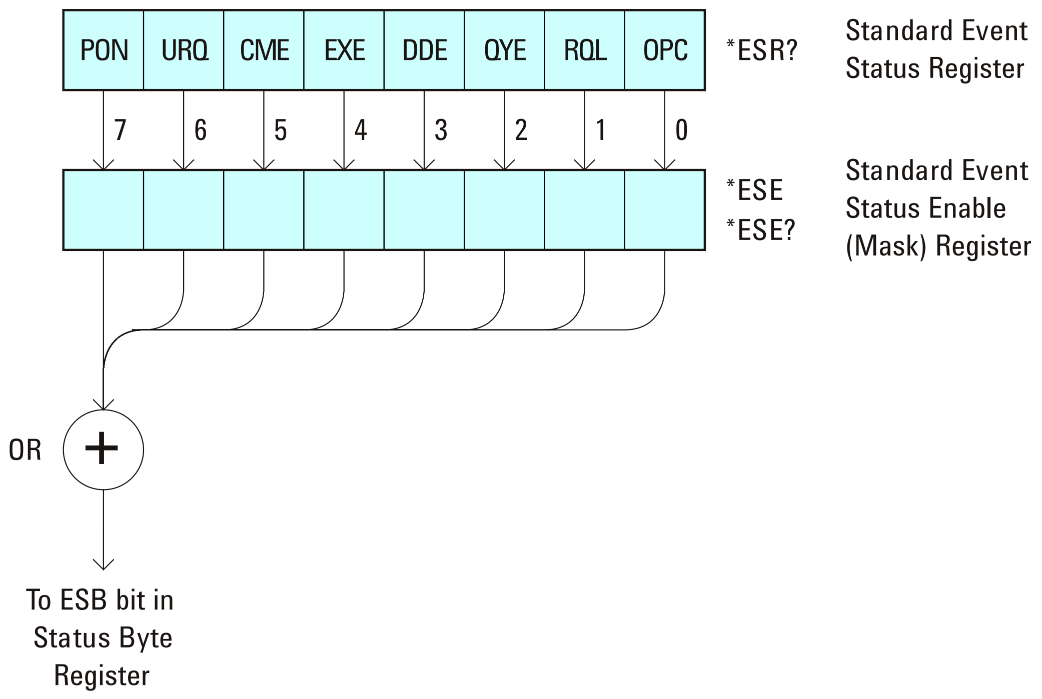

PON
URQ
CME
EXE
DDE
QYE
RQL
OPC
*ESR?
Standard Event Status Register
7
6
5
4
3
2
1
0
*ESE
*ESE?
Standard Event Status Enable (Mask) Register
OR
To ESB bit in Status Byte Register

Table 30 Standard Event Status Register (ESR)

Bit	Name	Description	When Set (1 = High = True), Indicates:
7	PON	Power On	An OFF to ON transition has occurred.
6	URQ	User Request	A front-panel key has been pressed.
5	CME	Command Error	A command error has been detected.
4	EXE	Execution Error	An execution error has been detected.
3	DDE	Device Dependent Error	A device-dependent error has been detected.
2	QYE	Query Error	A query error has been detected.

Table 30 Standard Event Status Register (ESR) (continued)

Bit	Name	Description	When Set (1 = High = True), Indicates:
1	RQL	Request Control	The device is requesting control. (Not used.)
0	OPC	Operation Complete	Operation is complete.

Return Format

```
<status><NL>

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

NOTE

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

See Also

- "Introduction to Common (*) Commands" on page 63
- "*ESE (Standard Event Status Enable)" on page 66
- "*OPC (Operation Complete)" on page 72
- "*CLS (Clear Status)" on page 65
- ":SYSTem:ERRor" on page 297

*IDN (Identification Number)

C (see page 530)

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 63
- "*OPT (Option Identification)" on page 73

*LRN (Learn Device Setup)

C (see page 530)

Query Syntax

```
*LRN?
```

The *LRN? query result contains the current state of the instrument. This query is similar to the :SYSTem:SETup? (see page 299) 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 63
- "*RCL (Recall)" on page 74
- "*SAV (Save)" on page 78
- ":SYSTem:SETup" on page 299

*OPC (Operation Complete)

C (see page 530)

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 63
- "*ESE (Standard Event Status Enable)" on page 66
- "*ESR (Standard Event Status Register)" on page 68
- "*CLS (Clear Status)" on page 65

*OPT (Option Identification)

C (see page 530)

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>,<reserved>,<reserved>,
                   <reserved>,<reserved>,<Low Speed Serial>,
                   <Automotive Serial>,<reserved>,<Secure>,<reserved>,
                   <reserved>,<reserved>,<reserved>,
                   <RS-232/UART Serial>,<reserved>

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

<reserved> ::= 0

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

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

<Secure> ::= {0 | SEC}

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

The *OPT? query returns the following:

Module	Module Id
No modules attached	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

See Also

- "Introduction to Common (*) Commands" on page 63
- "*IDN (Identification Number)" on page 70

*RCL (Recall)

C (see page 530)

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 63
- "*SAV (Save)" on page 78

*RST (Reset)

C (see page 530)

Command Syntax

```
*RST
```

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

Acquire Menu	
Mode	Normal
Realtime	On
Averaging	Off
# Averages	8

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

Cursor Menu	
Source	Channel 1

Display Menu	
Definite persistence	Off
Grid	33%
Vectors	On

Quick Meas Menu	
Source	Channel 1

Run Control	
	Scope is running

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

Trigger Menu	
Type	Edge
Mode	Auto
Coupling	dc
Source	Channel 1
Level	0.0 V
Slope	Positive
HF Reject and noise reject	Off
Holdoff	60 ns
External probe attenuation	AutoProbe (if AutoProbe is connected), otherwise 1.0:1

Trigger Menu	
External Units	Volts
External Impedance	1 M Ohm

See Also

- "Introduction to Common (*) Commands" on page 63

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 578

*SAV (Save)

C (see page 530)

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 63
- "*RCL (Recall)" on page 74

*SRE (Service Request Enable)

C (see page 530)

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.

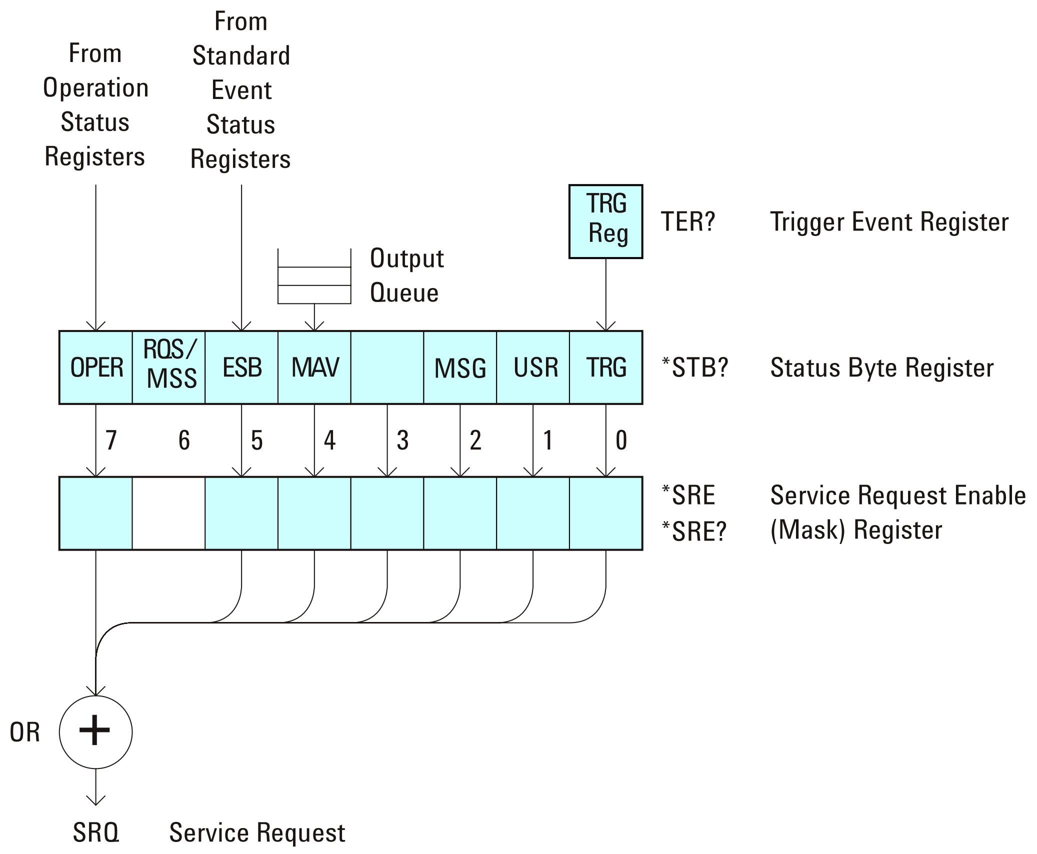


Table 31 Service Request Enable Register (SRE)

Bit	Name	Description	When Set (1 = High = True), Enables:
7	OPER	Operation Status Register	Interrupts when enabled conditions in the Operation Status Register (OPER) occur.
6	---	---	(Not used.)

Table 31 Service Request Enable Register (SRE) (continued)

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

Query Syntax

```
*SRE?
```

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

Return Format

```
<mask><NL>

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

See Also

- "Introduction to Common (*) Commands" on page 63
- "*STB (Read Status Byte)" on page 81
- "*CLS (Clear Status)" on page 65

*STB (Read Status Byte)

C (see page 530)

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`

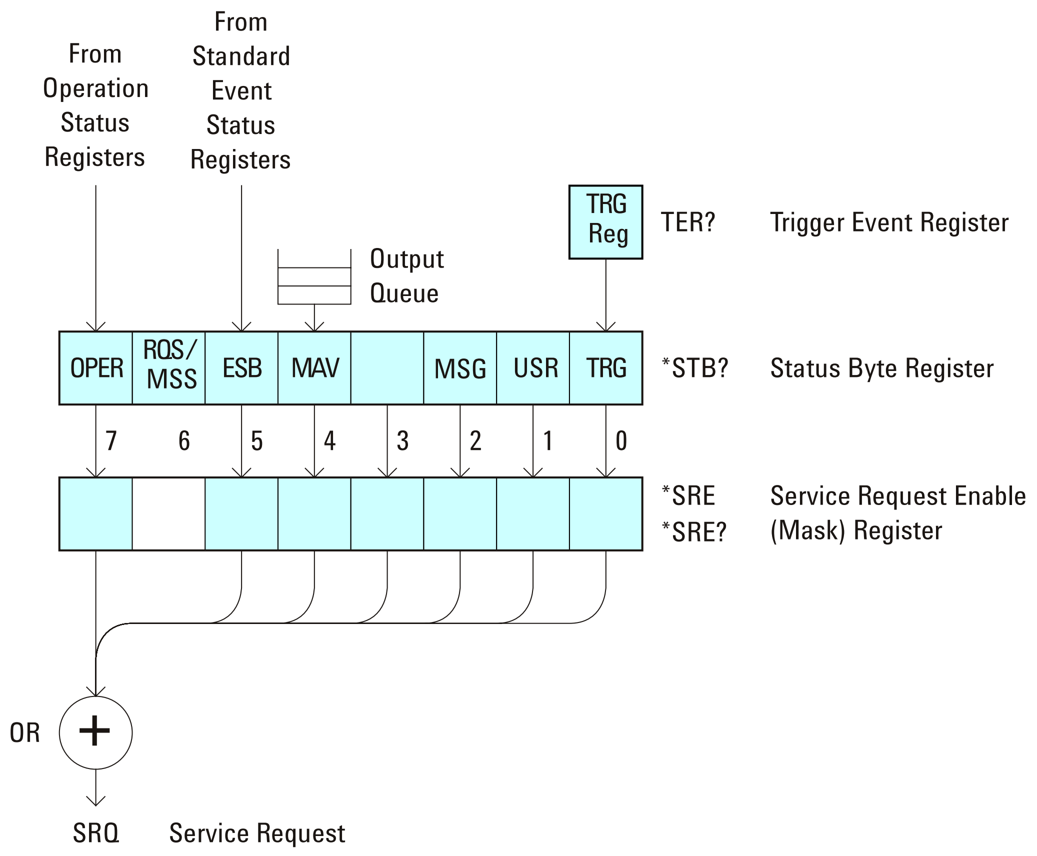


Table 32 Status Byte Register (STB)

Bit	Name	Description	When Set (1 = High = True), Indicates:
7	OPER	Operation Status Register	An enabled condition in the Operation Status Register (OPER) has occurred.

Table 32 Status Byte Register (STB) (continued)

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

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

See Also
- "Introduction to Common (*) Commands" on page 63
- "*SRE (Service Request Enable)" on page 79

*TRG (Trigger)

C (see page 530)

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 63
- ":DIGitize" on page 96
- ":RUN" on page 116
- ":STOP" on page 120

*TST (Self Test)

C (see page 530)

Query Syntax

```
*TST?
```

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

Return Format

```
<result><NL>

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

See Also

- "Introduction to Common (*) Commands" on page 63

*WAI (Wait To Continue)

C (see page 530)

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 63

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 88.

Table 33 Root (:) Commands Summary

Command	Query	Options and Query Returns
n/a	:AER? (see page 89)	{0 | 1}; an integer in NR1 format
:AUToscale [<source>[,..,<source>]] (see page 90)	n/a	<source> ::= CHANnel<n> <source> can be repeated up to 5 times <n> ::= 1-2 or 1-4 in NR1 format
:AUToscale:AMODE <value> (see page 92)	:AUToscale:AMODE? (see page 92)	<value> ::= {NORMal | CURRent}}
:AUToscale:CHANnels <value> (see page 93)	:AUToscale:CHANnels? (see page 93)	<value> ::= {ALL | DISPlayed}}
:BLANk [<source>] (see page 94)	n/a	<source> ::= {CHANnel<n>} | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:CDISplay (see page 95)	n/a	n/a
:DIGitize [<source>[,..,<source>]] (see page 96)	n/a	<source> ::= {CHANnel<n> | FUNCtion | MATH} <source> can be repeated up to 5 times <n> ::= 1-2 or 1-4 in NR1 format
:HWEenable <n> (see page 98)	:HWEenable? (see page 98)	<n> ::= 16-bit integer in NR1 format
n/a	:HWERregister:CONDition? (see page 100)	<n> ::= 16-bit integer in NR1 format
n/a	:HWERegister[:EVENt]? (see page 102)	<n> ::= 16-bit integer in NR1 format
:MERGe <pixel memory> (see page 104)	n/a	<pixel memory> ::= {PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}}
:OPEE <n> (see page 105)	:OPEE? (see page 106)	<n> ::= 16-bit integer in NR1 format
n/a	:OPERregister:CONDition? (see page 107)	<n> ::= 16-bit integer in NR1 format
n/a	:OPERegister[:EVENt]? (see page 109)	<n> ::= 16-bit integer in NR1 format

Table 33 Root (:) Commands Summary (continued)

Command	Query	Options and Query Returns
:OVLenable <mask> (see page 111)	:OVLenable? (see page 112)	<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	:OVLRegister? (see page 113)	<value> ::= integer in NR1 format. See OVLenable for <value>
:PRINt [<options>] (see page 115)	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 116)	n/a	n/a
n/a	:SERial (see page 117)	<return value> ::= unquoted string containing serial number
:SINGle (see page 118)	n/a	n/a

n/a	:STATus? <display> (see page 119)	{0 | 1}
		<display> ::= {CHANnel<n> | FUNCtion | MATH}
		<n> ::= 1-2 or 1-4 in NR1 format

:STOP (see page 120)	n/a	n/a
n/a	:TER? (see page 121)	{0 | 1}
:VIEW <source> (see page 122)	n/a	<source> ::= {CHANnel<n> | PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format

Introduction to Root (:) Commands

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

:AER (Arm Event Register)

C (see page 530)

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 88
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 107
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:AUToscale

C (see page 530)

Command Syntax

```
:AUToscale

:AUToscale [<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 <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 93) 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.
- 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*.

See Also

- "Introduction to Root (:) Commands" on page 88
- ":AUToscale:CHANnels" on page 93

- ":AUToscale:AMODE" on page 92

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 578

:AUToscale:AMODE

N (see page 530)

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 88
- ":AUToscale" on page 90
- ":AUToscale:CHANnels" on page 93
- ":ACQuire:TYPE" on page 132
- ":ACQuire:MODE" on page 129

:AUToscale:CHANnels

N (see page 530)

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 88
- ":AUToscale" on page 90
- ":AUToscale:AMODE" on page 92
- ":VIEW" on page 122
- ":BLANk" on page 94

:BLANk

N (see page 530)

Command Syntax

```
:BLANk [<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 :BLANk command turns off (stops displaying) the specified channel or math function. 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 or :FUNCtion: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 88
- ":CDISplay" on page 95
- ":CHANnel<n>:DISPlay" on page 147
- ":FUNCtion:DISPlay" on page 184
- ":STATus" on page 119
- ":VIEW" on page 122

Example Code

- "Example Code" on page 122

:CDISplay

C (see page 530)

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 88
- ":DISPlay:CLEar" on page 163

:DIGitize

C (see page 530)

Command Syntax

```
:DIGitize [<source>[,..,<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 <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 88
- ":RUN" on page 116
- ":SINGle" on page 118
- ":STOP" on page 120
- ":ACQuire Commands" on page 123
- ":WAVeform Commands" on page 401

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 578

:HWEenable (Hardware Event Enable Register)

N (see page 530)

Command Syntax

```
:HWEenable <mask>

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

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

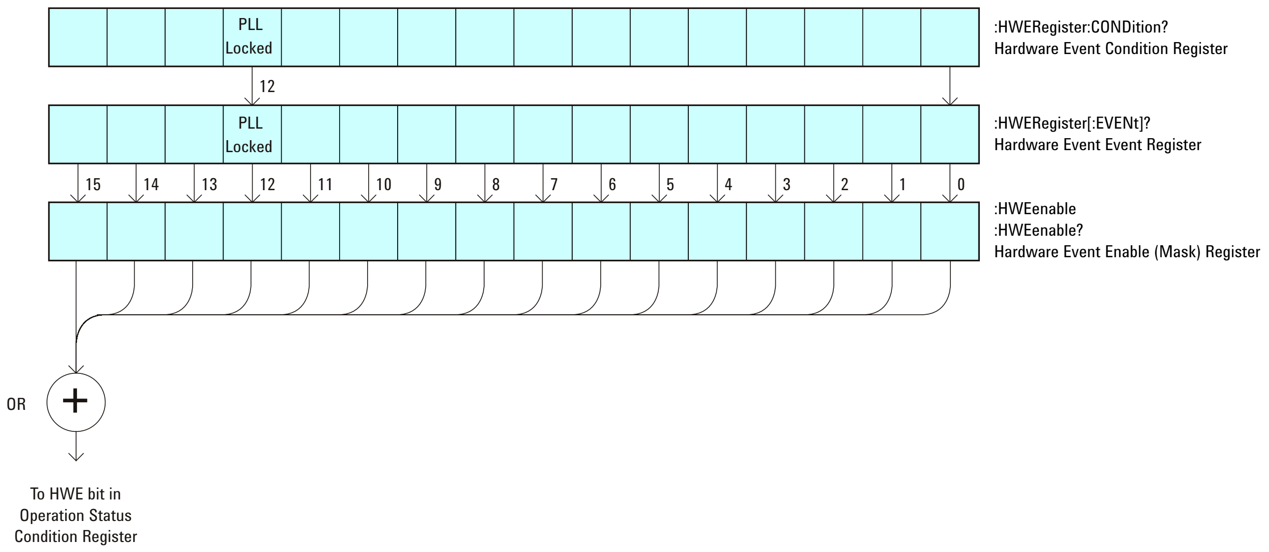

PLL Locked
:HWERegister:CONDition?
Hardware Event Condition Register
12
PLL Locked
:HWERegister[:EVENt]?
Hardware Event Event Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
:HWEenable
:HWEenable?
Hardware Event Enable (Mask) Register
OR
To HWE bit in Operation Status Condition Register

Table 34 Hardware Event Enable Register (HWEenable)

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

Query Syntax

```
:HWEenable?
```

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88

- ":AER (Arm Event Register)" on page 89
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- ":OVLenable (Overload Event Enable Register)" on page 111
- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:HWERegister:CONDition (Hardware Event Condition Register)

N (see page 530)

Query Syntax :HWERegister:CONDition?

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

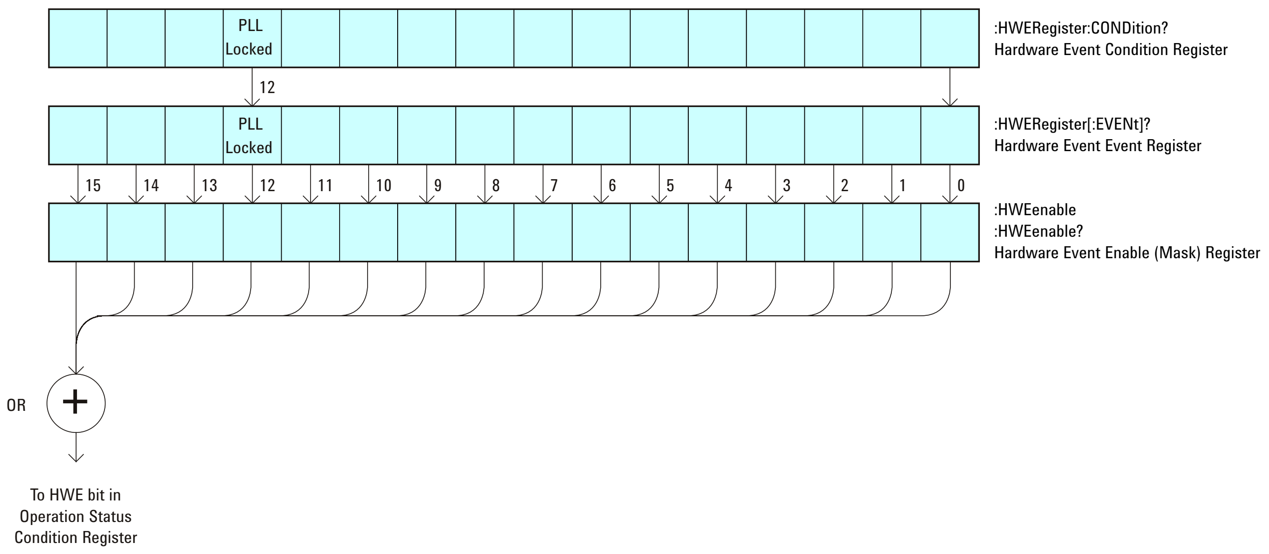


Table 35 Hardware Event Condition Register

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

Return Format `<value><NL>`

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- ":OVLenable (Overload Event Enable Register)" on page 111

- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:HWERegister[:EVENt] (Hardware Event Event Register)

N (see page 530)

Query Syntax

```
:HWERegister[:EVENt]?
```

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

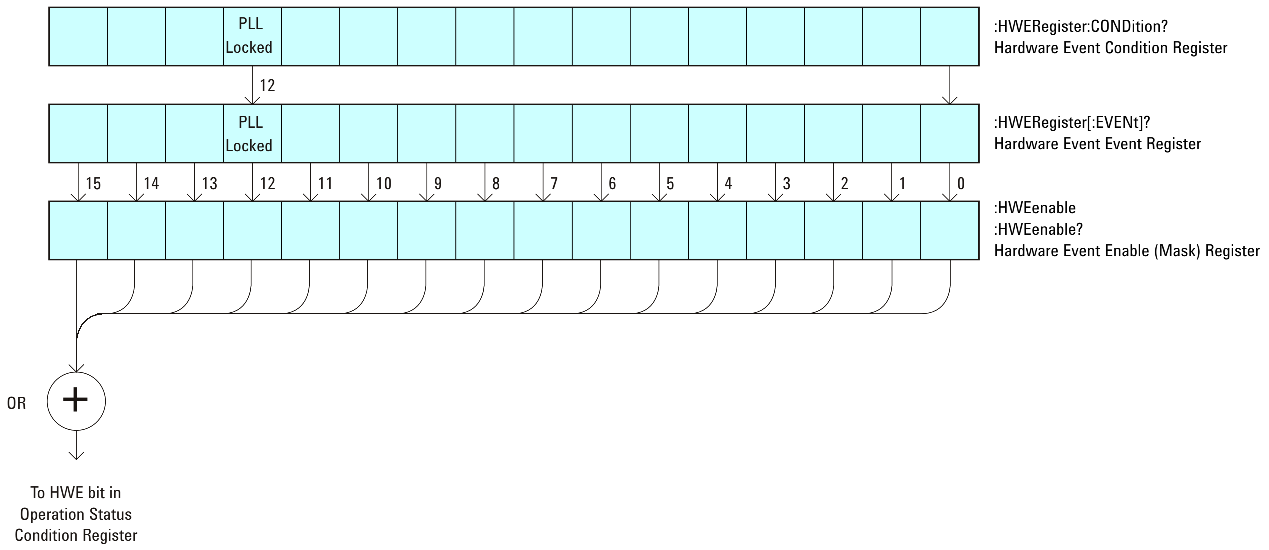


Table 36 Hardware Event Event Register

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 107
- ":OVLenable (Overload Event Enable Register)" on page 111

- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:MERGe

N (see page 530)

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 88
- "*SAV (Save)" on page 78
- "*RCL (Recall)" on page 74
- ":VIEW" on page 122
- ":BLANk" on page 94

:OPEE (Operation Status Enable Register)

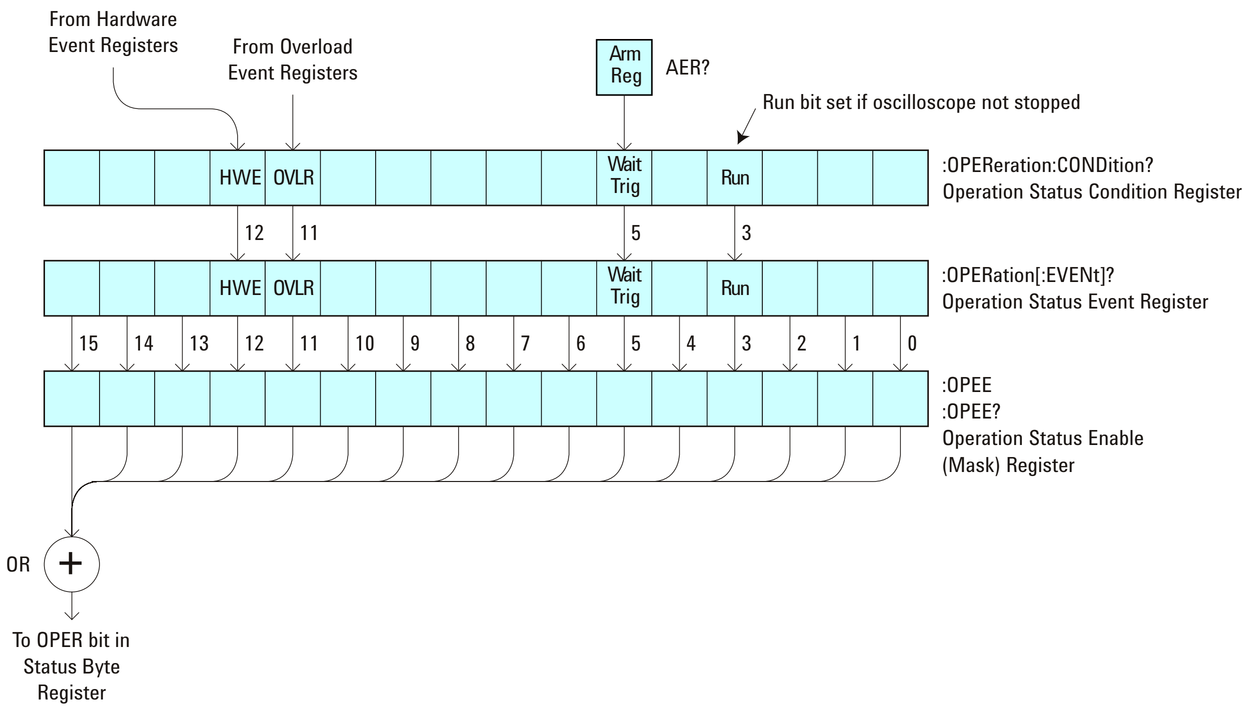
C (see page 530)

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.



From Hardware Event Registers
From Overload Event Registers
Arm Reg
AER?
Run bit set if oscilloscope not stopped
HWE
OVLR
Wait Trig
Run
:OPERation:CONDition?
Operation Status Condition Register
12
11
5
3
:OPERation[:EVENt]?
Operation Status Event Register
15
14
13
10
9
8
7
6
4
2
1
0
:OPEE
:OPEE?
Operation Status Enable (Mask) Register
OR
To OPER bit in Status Byte Register

Table 37 Operation Status Enable Register (OPEE)

Bit	Name	Description	When Set (1 = High = True), Enables:
15-13	---	---	(Not used.)
12	HWE	Hardware Event	Event when hardware event occurs.
11	OVLR	Overload	Event when 50Ω input overload occurs.
10-6	---	---	(Not used.)
5	Wait Trig	Wait Trig	Event when the trigger is armed.
4	---	---	(Not used.)

Table 37 Operation Status Enable Register (OPEE) (continued)

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

Query Syntax

```
:OPEE?
```

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":AER (Arm Event Register)" on page 89
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- ":OVLenable (Overload Event Enable Register)" on page 111
- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:OPERegister:CONDition (Operation Status Condition Register)

C (see page 530)

Query Syntax `:OPERegister:CONDition?`

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

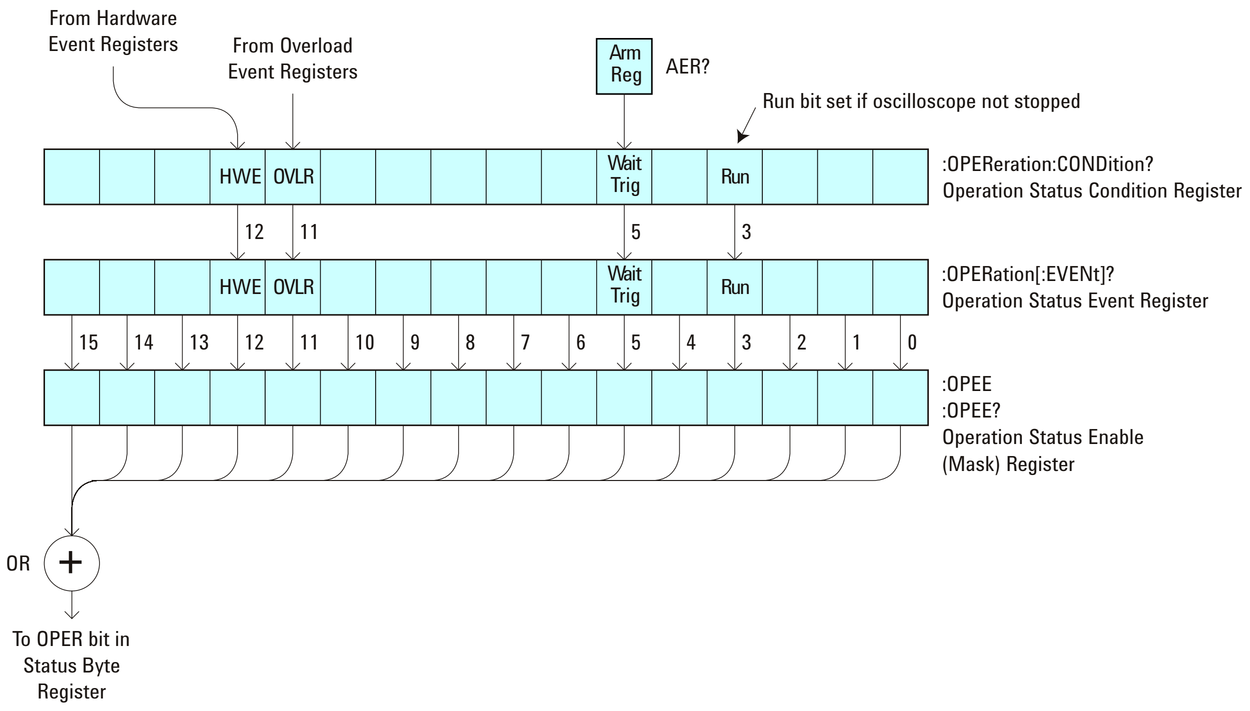

From Hardware Event Registers
From Overload Event Registers
Arm Reg
AER?
Run bit set if oscilloscope not stopped
HWE
OVLR
Wait Trig
Run
:OPERation:CONDition?
Operation Status Condition Register
12
11
5
3
:OPERation[:EVENt]?
Operation Status Event Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
:OPEE
:OPEE?
Operation Status Enable (Mask) Register
OR
To OPER bit in Status Byte Register

Table 38 Operation Status Condition Register

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- ":OVLenable (Overload Event Enable Register)" on page 111
- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79
- ":HWERegister[:EVENt] (Hardware Event Event Register)" on page 102
- ":HWEenable (Hardware Event Enable Register)" on page 98

:OPERegister[:EVENt] (Operation Status Event Register)

C (see page 530)

Query Syntax :OPERegister[:EVENt]?

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

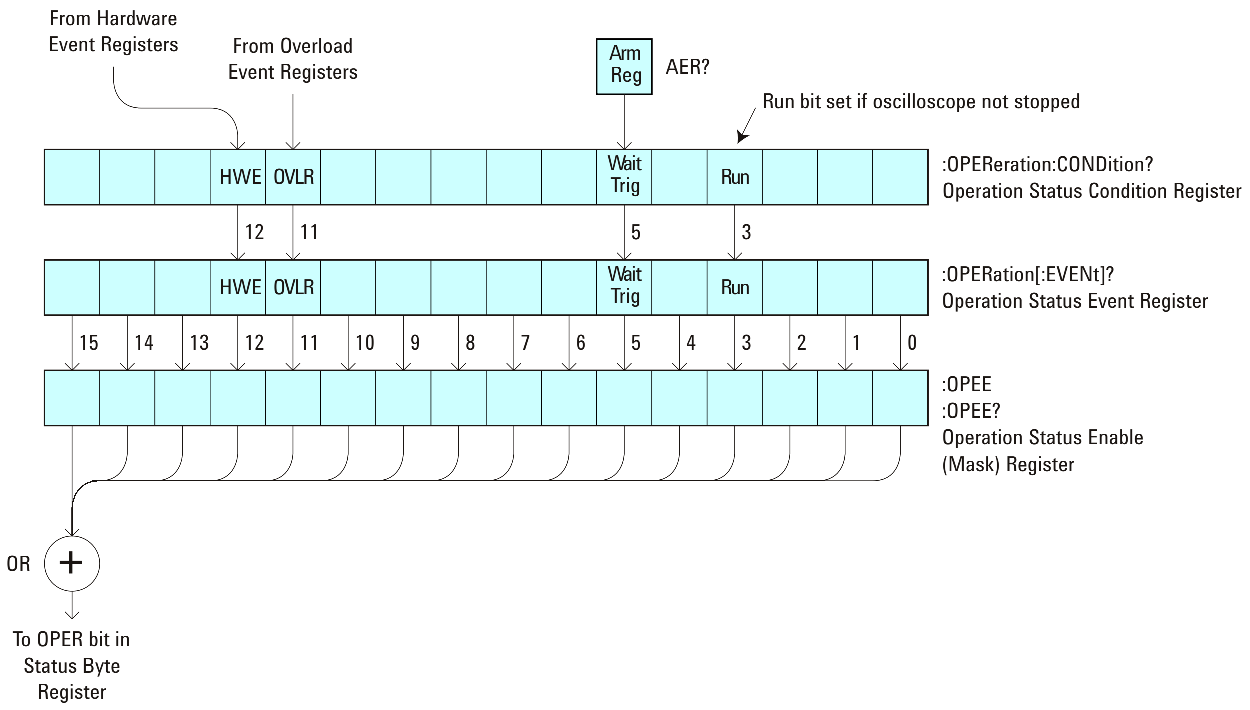

From Hardware Event Registers
From Overload Event Registers
Arm Reg
AER?
Run bit set if oscilloscope not stopped
HWE
OVLR
Wait Trig
Run
:OPERation:CONDition?
Operation Status Condition Register
12
11
5
3
:OPERation[:EVENt]?
Operation Status Event Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
:OPEE
:OPEE?
Operation Status Enable (Mask) Register
OR
To OPER bit in Status Byte Register

Table 39 Operation Status Event Register

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 107
- ":OVLenable (Overload Event Enable Register)" on page 111
- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79
- ":HWERegister[:EVENt] (Hardware Event Event Register)" on page 102
- ":HWEenable (Hardware Event Enable Register)" on page 98

:OVLenable (Overload Event Enable Register)

C (see page 530)

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. If there are only two analog channels, you can also set external trigger input impedance to 50Ω.

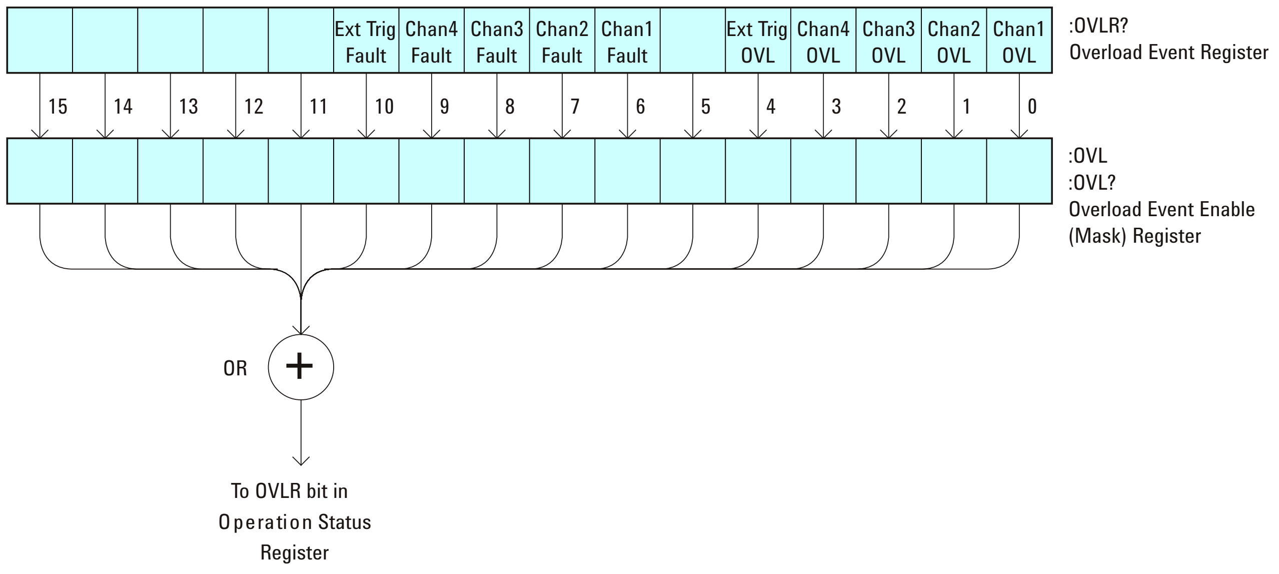


Table 40 Overload Event Enable Register (OVL)

Bit	Description	When Set (1 = High = True), Enables:
15-11	---	(Not used.)
10	External Trigger Fault	Event when fault occurs on External Trigger input.
9	Channel 4 Fault	Event when fault occurs on Channel 4 input.
8	Channel 3 Fault	Event when fault occurs on Channel 3 input.

Table 40 Overload Event Enable Register (OVL) (continued)

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

Query Syntax

```
:OVLenable?
```

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

Return Format

```
<enable_mask><NL>
```

```
<enable_mask> ::= integer in NR1 format.
```

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 107
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- ":OVLRegister (Overload Event Register)" on page 113
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:OVLRegister (Overload Event Register)

C (see page 530)

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. If there are only two analog channels, you can also set external trigger input impedance to 50Ω.

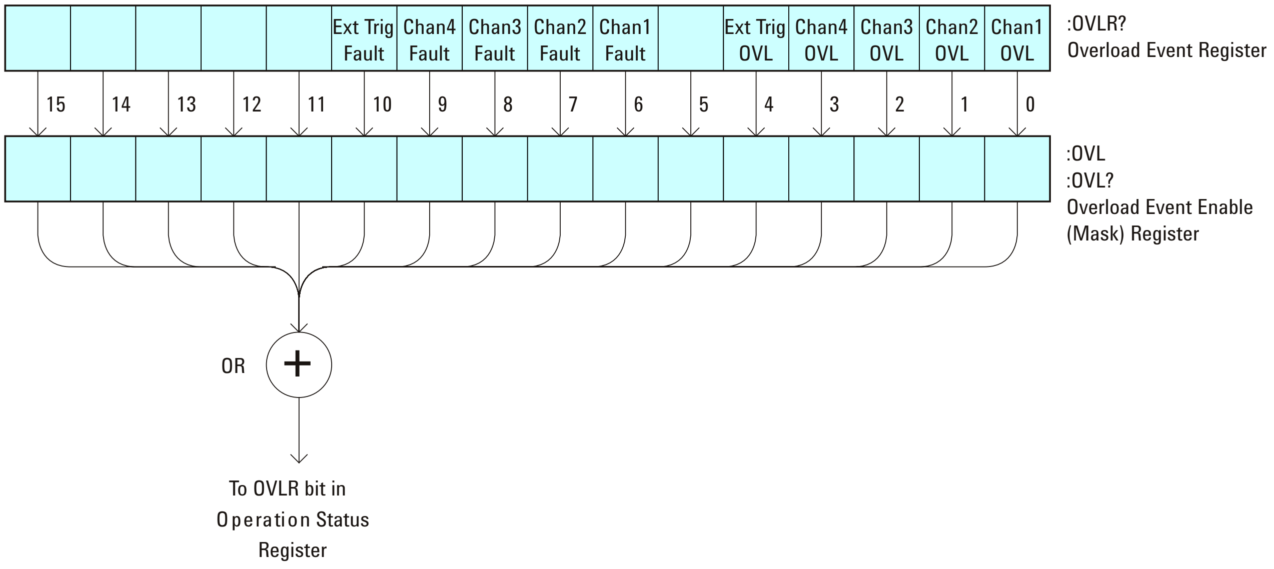


Table 41 Overload Event Register (OVLR)

Bit	Description	When Set (1 = High = True), Indicates:
15-11	---	(Not used.)
10	External Trigger Fault	Fault has occurred on External Trigger input.
9	Channel 4 Fault	Fault has occurred on Channel 4 input.
8	Channel 3 Fault	Fault has occurred on Channel 3 input.
7	Channel 2 Fault	Fault has occurred on Channel 2 input.
6	Channel 1 Fault	Fault has occurred on Channel 1 input.
5	---	(Not used.)

Table 41 Overload Event Register (OVLR) (continued)

Bit	Description	When Set (1 = High = True), Indicates:
4	External Trigger OVL	Overload has occurred on External Trigger input.
3	Channel 4 OVL	Overload has occurred on Channel 4 input.
2	Channel 3 OVL	Overload has occurred on Channel 3 input.
1	Channel 2 OVL	Overload has occurred on Channel 2 input.
0	Channel 1 OVL	Overload has occurred on Channel 1 input.

Return Format

```
<value><NL>

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":CHANnel<n>:PROTection" on page 156
- ":EXTernal:PROTection" on page 178
- ":OPEE (Operation Status Enable Register)" on page 105
- ":OVLenable (Overload Event Enable Register)" on page 111
- "*STB (Read Status Byte)" on page 81
- "*SRE (Service Request Enable)" on page 79

:PRINt

C (see page 530)

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 474 for more information.

See Also

- "Introduction to Root (:) Commands" on page 88
- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:FORMat" on page 474
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:GRAYscale" on page 475
- ":DISPlay:DATA" on page 164

:RUN

C (see page 530)

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 88
- ":SINGle" on page 118
- ":STOP" on page 120

Example Code

```
' RUN_STOP - (not executed in this example)
'  - RUN starts the data acquisition for the active waveform display.
'  - STOP stops the data acquisition and turns off AUTOSTORE.
' myScope.WriteString ":RUN"    ' Start data acquisition.
' myScope.WriteString ":STOP"    ' Stop the data acquisition.
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:SERial

N (see page 530)

Query Syntax

```
:SERial?
```

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

Return Format:

```
Unquoted string<NL>
```

See Also

- "Introduction to Root (:) Commands" on page 88

:SINGle

C (see page 530)

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 88
- ":RUN" on page 116
- ":STOP" on page 120

:STATus

N (see page 530)

Query Syntax

```
:STATus? <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 :STATus? query reports whether the channel or function specified by <source> is displayed.

NOTE MATH is an alias for FUNCtion.

Return Format

```
<value><NL>
```

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

See Also

- "Introduction to Root (:) Commands" on page 88
- ":BLANk" on page 94
- ":CHANnel<n>:DISPlay" on page 147
- ":FUNCtion:DISPlay" on page 184
- ":VIEW" on page 122

:STOP

C (see page 530)

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 88
- ":RUN" on page 116
- ":SINGle" on page 118

Example Code

- "Example Code" on page 116

:TER (Trigger Event Register)

C (see page 530)

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 88
- "*SRE (Service Request Enable)" on page 79
- "*STB (Read Status Byte)" on page 81

:VIEW

N (see page 530)

Command Syntax

```
:VIEW <source>

<source> ::= {CHANnel<n> | PMEMory0,..,PMEMory9 | FUNCtion | MATH}

<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, or trace memory.

NOTE MATH is an alias for FUNCtion.

See Also

- "Introduction to Root (:) Commands" on page 88
- ":BLANk" on page 94
- ":CHANnel<n>:DISPlay" on page 147
- ":FUNCtion:DISPlay" on page 184
- ":STATus" on page 119

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 578

:ACQuire Commands

Set the parameters for acquiring and storing data. See "Introduction to :ACQuire Commands" on page 123.

Table 42 :ACQuire Commands Summary

Command	Query	Options and Query Returns
n/a	:ACQuire:AALias? (see page 125)	{1 | 0}
:ACQuire:COMPlete <complete> (see page 126)	:ACQuire:COMPlete? (see page 126)	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNt <count> (see page 127)	:ACQuire:COUNt? (see page 127)	<count> ::= an integer from 1 to 65536 in NR1 format
:ACQuire:DAALias <mode> (see page 128)	:ACQuire:DAALias? (see page 128)	<mode> ::= {DISable | AUTO}
:ACQuire:MODE <mode> (see page 129)	:ACQuire:MODE? (see page 129)	<mode> ::= {RTIMe | ETIMe}
n/a	:ACQuire:POINts? (see page 130)	<# points> ::= an integer in NR1 format
n/a	:ACQuire:SRATe? (see page 131)	<sample_rate> ::= sample rate (samples/s) in NR3 format
:ACQuire:TYPE <type> (see page 132)	:ACQuire:TYPE? (see page 132)	<type> ::= {NORMal | AVERage | HRESolution | PEAK}

Introduction to :ACQuire Commands

The ACQuire subsystem controls the way in which waveforms are acquired. These acquisition types are available: normal, averaging, peak detect, and high resolution. 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
```

:ACQuire:AALias

N (see page 530)

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 123
- ":ACQuire:DAALias" on page 128

:ACQuire:COMPlete

C (see page 530)

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 123
- ":ACQuire:TYPE" on page 132
- ":DIGitize" on page 96
- ":WAVeform:POINts" on page 413

Example Code

```
  ' AQUIRE_COMPLETE - Specifies the minimum completion criteria for
  ' an acquisition.  The parameter determines the percentage of time
  ' buckets needed to be "full" before an acquisition is considered
  ' to be complete.
  myScope.WriteString ":ACQUIRE:COMPLETE 100"
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:ACQuire:COUNt

C (see page 530)

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 123
- ":ACQuire:TYPE" on page 132
- ":DIGitize" on page 96
- ":WAVeform:COUNt" on page 409

:ACQuire:DAALias

N (see page 530)

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 123
- ":ACQuire:AALias" on page 125

:ACQuire:MODE

C (see page 530)

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 123
- ":ACQuire:TYPE" on page 132

:ACQuire:POINts

C (see page 530)

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 123
- ":DIGitize" on page 96
- ":WAVeform:POINts" on page 413

:ACQuire:SRATe

N (see page 530)

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 123
- ":ACQuire:POINts" on page 130

:ACQuire:TYPE

C (see page 530)

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 123

- ":ACQuire:COUNt" on page 127
- ":ACQuire:MODE" on page 129
- ":DIGitize" on page 96
- ":WAVeform:TYPE" on page 425
- ":WAVeform:PREamble" on page 417

Example Code

```
' AQUIRE_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 578

:CALibrate Commands

Utility commands for viewing calibration status and for starting the user calibration procedure. See "Introduction to :CALibrate Commands" on page 134.

Table 43 :CALibrate Commands Summary

Command	Query	Options and Query Returns
n/a	:CALibrate:DATE? (see page 135)	<return value> ::= <day>,<month>,<year>; all in NR1 format
:CALibrate:LABel <string> (see page 136)	:CALibrate:LABel? (see page 136)	<string> ::= quoted ASCII string up to 32 characters
:CALibrate:STARt (see page 137)	n/a	n/a
n/a	:CALibrate:STATus? (see page 138)	<return value> ::= ALL,<status_code>,<status_string> <status_code> ::= an integer status code <status_string> ::= an ASCII status string
n/a	:CALibrate:SWITch? (see page 139)	{PROTected | UNPRotected}
n/a	:CALibrate:TEMPerature? (see page 140)	<return value> ::= degrees C delta since last cal in NR3 format
n/a	:CALibrate:TIME? (see page 141)	<return value> ::= <hours>,<minutes>,<seconds>; all in NR1 format

Introduction to :CALibrate Commands

The CALibrate subsystem provides utility commands for:

- Determining the state of the calibration factor protection switch (CAL PROTECT).
- Saving and querying the calibration label string.
- Reporting the calibration time and date.
- Reporting changes in the temperature since the last calibration.
- Starting the user calibration procedure.

:CALibrate:DATE

N (see page 530)

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 134

:CALibrate:LABel

N (see page 530)

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 134

:CALibrate:STARt

N (see page 530)

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 134
- ":CALibrate:SWITch" on page 139

:CALibrate:STATus

N (see page 530)

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 134

:CALibrate:SWITch

N (see page 530)

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 134

:CALibrate:TEMPerature

N (see page 530)

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 134

:CALibrate:TIME

N (see page 530)

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 134

:CHANnel<n> Commands

Control all oscilloscope functions associated with individual analog channels or groups of channels. See "Introduction to :CHANnel<n> Commands" on page 143.

Table 44 :CHANnel<n> Commands Summary

Command	Query	Options and Query Returns
:CHANnel<n>:BWLimit {{0 | OFF} | {1 | ON}} (see page 145)	:CHANnel<n>:BWLimit? (see page 145)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:COUPling <coupling> (see page 146)	:CHANnel<n>:COUPling? (see page 146)	<coupling> ::= {AC | DC} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:DISPlay {{0 | OFF} | {1 | ON}} (see page 147)	:CHANnel<n>:DISPlay? (see page 147)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:IMPedance <impedance> (see page 148)	:CHANnel<n>:IMPedance? (see page 148)	<impedance> ::= {ONEMeg | FIFTy} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:INVert {{0 | OFF} | {1 | ON}} (see page 149)	:CHANnel<n>:INVert? (see page 149)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:LABel <string> (see page 150)	:CHANnel<n>:LABel? (see page 150)	<string> ::= any series of 6 or less ASCII characters enclosed in quotation marks <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:OFFSet <offset>[suffix] (see page 151)	:CHANnel<n>:OFFSet? (see page 151)	<offset> ::= Vertical offset value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe <attenuation> (see page 152)	:CHANnel<n>:PROBe? (see page 152)	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4r in NR1 format
n/a	:CHANnel<n>:PROBe:ID? (see page 153)	<probe id> ::= unquoted ASCII string up to 11 characters <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:PROBe:SKEW <skew_value> (see page 154)	:CHANnel<n>:PROBe:SKEW? (see page 154)	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1-2 or 1-4 in NR1 format

Table 44 :CHANnel<n> Commands Summary (continued)

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe:STYPe <signal type> (see page 155)	:CHANnel<n>:PROBe:STYPe? (see page 155)	<signal type> ::= {DIFFerential | SINGle} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:PROTection (see page 156)	:CHANnel<n>:PROTection? (see page 156)	{NORM | TRIP} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:RANGe <range>[suffix] (see page 157)	:CHANnel<n>:RANGe? (see page 157)	<range> ::= Vertical full-scale range value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:SCALe <scale>[suffix] (see page 158)	:CHANnel<n>:SCALe? (see page 158)	<scale> ::= Vertical units per division value in NR3 format [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:UNITs <units> (see page 159)	:CHANnel<n>:UNITs? (see page 159)	<units> ::= {VOLT | AMPere} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:VERNier {{0 | OFF} | {1 | ON}} (see page 160)	:CHANnel<n>:VERNier? (see page 160)	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format

Introduction to :CHANnel<n> Commands

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

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

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

Use :CHANnel1?, :CHANnel2?, :CHANnel3? or :CHANnel4? to query setup information for the CHANnel<n> subsystem.

Return Format

The following are sample responses from the :CHANnel<n>? query. In this case, the query was issued following a *RST command.

```
:CHAN1:RANG +40.0E+00;OFFS +0.00000E+00;COUP DC;IMP ONEM;DISP 1;BWL 0;
INV 0;LAB "1";UNIT VOLT;PROB +10E+00;PROB:SKEW +0.00E+00;STYP SING
```

:CHANnel<n>:BWLimit

C (see page 530)

Command Syntax

```
:CHANnel<n>:BWLimit <bwlimit>
```

```
<bwlimit> ::= {{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 143

:CHANnel<n>:COUPling

C (see page 530)

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 143

:CHANNel<n>:DISPlay

C (see page 530)

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 143
- ":VIEW" on page 122
- ":BLANk" on page 94
- ":STATus" on page 119

:CHANnel<n>:IMPedance

C (see page 530)

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).

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 143

:CHANnel<n>:INVert

N (see page 530)

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 143

:CHANnel<n>:LABel

N (see page 530)

Command Syntax

```
:CHANnel<n>:LABel <string>

<string> ::= quoted ASCII string

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

<n> ::= {1 | 2} for the two channel oscilloscope 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. Lower case characters are converted to upper case.

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

Query Syntax

```
:CHANnel<n>:LABel?
```

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

Return Format

```
<string><NL>

<string> ::= quoted ASCII string
```

See Also

- "Introduction to :CHANnel<n> Commands" on page 143
- ":DISPlay:LABel" on page 166
- ":DISPlay:LABList" on page 167

Example Code

```
   ' 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 ":CHANNEL1:LABEL ""CAL 1"""    ' Label channel1 "C
AL 1".
   myScope.WriteString ":CHANNEL2:LABEL ""CAL2"""    ' Label channel1 "CA
L2".
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:CHANnel<n>:OFFSet

C (see page 530)

Command Syntax

```
:CHANnel<n>:OFFSet <offset> [<suffix>]

<offset> ::= Vertical offset 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>:OFFSet command sets the value that is represented at center screen for the selected channel. The range of legal values varies with the value set by the :CHANnel<n>:RANGe and :CHANnel<n>:SCALe commands. If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value. Legal values are affected by the probe attenuation setting.

Query Syntax

```
:CHANnel<n>:OFFSet?
```

The :CHANnel<n>:OFFSet? query returns the current offset value for the selected channel.

Return Format

```
<offset><NL>

<offset> ::= Vertical offset value in NR3 format
```

See Also

- "Introduction to :CHANnel<n> Commands" on page 143
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:SCALe" on page 158
- ":CHANnel<n>:PROBe" on page 152

:CHANnel<n>:PROBe

C (see page 530)

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 143
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:SCALe" on page 158
- ":CHANnel<n>:OFFSet" on page 151

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 578

:CHANnel<n>:PROBe:ID

C (see page 530)

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 143

:CHANnel<n>:PROBe:SKEW

C (see page 530)

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 143

:CHANnel<n>:PROBe:STYPe

C (see page 530)

Command Syntax

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

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

<signal type> ::= {DIFFerential | SINGle}

<n> ::= {1 | 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 143
- ":CHANnel<n>:OFFSet" on page 151

:CHANnel<n>:PROTection

N (see page 530)

Command Syntax

```
:CHANnel<n>:PROTection[:CLEar]
```

```
<n> ::= {1 | 2 | 3 | 4}
```

When the analog channel input impedance is set to 50Ω, 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 148) 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 143
- ":CHANnel<n>:COUPling" on page 146
- ":CHANnel<n>:IMPedance" on page 148
- ":CHANnel<n>:PROBe" on page 152

:CHANnel<n>:RANGe

C (see page 530)

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:

- 16 mV to 40 V.

If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

Query Syntax

```
:CHANnel<n>:RANGe?
```

The :CHANnel<n>:RANGe? query returns the current full-scale range setting for the specified channel.

Return Format

```
<range_argument><NL>
```

```
<range_argument> ::= vertical full-scale range value in NR3 format
```

See Also

- "Introduction to :CHANnel<n> Commands" on page 143
- ":CHANnel<n>:SCALe" on page 158
- ":CHANnel<n>:PROBe" on page 152

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 578

:CHANnel<n>:SCALe

N (see page 530)

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:

- 2 mV to 5 V.

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 143
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:PROBe" on page 152

:CHANnel<n>:UNITs

N (see page 530)

Command Syntax

```
:CHANnel<n>:UNITs <units>

<units> ::= {VOLT | AMPere}

<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 VOLT for a voltage probe and select AMPere for a current probe. Measurement results, channel sensitivity, and trigger level will reflect the measurement units you select.

Query Syntax

```
:CHANnel<n>:UNITs?
```

The :CHANnel<n>:UNITs? query returns the current units setting for the specified channel.

Return Format

```
<units><NL>

<units> ::= {VOLT | AMP}
```

See Also

- "Introduction to :CHANnel<n> Commands" on page 143
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:PROBe" on page 152
- ":EXTernal:UNITs" on page 180

:CHANnel<n>:VERNier

N (see page 530)

Command Syntax

```
:CHANnel<n>:VERNier <vernier value>
```

```
<vernier 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

```
<vernier value><NL>
```

```
<vernier value> ::= {0 | 1}
```

See Also

• "Introduction to :CHANnel<n> Commands" on page 143

:DISPlay Commands

Control how waveforms, graticule, and text are displayed and written on the screen. See "Introduction to :DISPlay Commands" on page 161.

Table 45 :DISPlay Commands Summary

Command	Query	Options and Query Returns
:DISPlay:CLEar (see page 163)	n/a	n/a
:DISPlay:DATA [<format>][,][<area>][,][<palette>]<display data> (see page 164)	:DISPlay:DATA? [<format>][,][<area>][,][<palette>] (see page 164)	<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 166)	:DISPlay:LABel? (see page 166)	{0 | 1}
:DISPlay:LABList <binary block> (see page 167)	:DISPlay:LABList? (see page 167)	<binary block> ::= an ordered list of up to 75 labels, each 6 characters maximum, separated by newline characters
:DISPlay:PERSistence <value> (see page 168)	:DISPlay:PERSistence? (see page 168)	<value> ::= {MINimum | INFinite}}
:DISPlay:SOURce <value> (see page 169)	:DISPlay:SOURce? (see page 169)	<value> ::= {PMEMory{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}}
:DISPlay:VECTors {{1 | ON} | {0 | OFF}} (see page 170)	:DISPlay:VECTors? (see page 170)	{1 | 0}

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

N (see page 530)

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 161
- ":CDISplay" on page 95

:DISPlay:DATA

N (see page 530)

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. The :HARDcopy:INKSaver setting also affects the screen data. 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.

NOTE 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 161
- ":DISPlay:SOURce" on page 169
- ":HARDcopy:INKSaver" on page 199
- ":MERGe" on page 104
- ":PRINt" on page 115
- "*RCL (Recall)" on page 74
- "*SAV (Save)" on page 78
- ":VIEW" on page 122

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 f
or 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 578

:DISPlay:LABel

N (see page 530)

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 channel labels.

Return Format

```
<value><NL>

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

See Also

- "Introduction to :DISPlay Commands" on page 161
- ":CHANnel<n>:LABel" on page 150

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 578

:DISPlay:LABList

N (see page 530)

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 alphabetical 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 161
- ":DISPlay:LABel" on page 166
- ":CHANnel<n>:LABel" on page 150

:DISPlay:PERSistence

N (see page 530)

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 161
- ":DISPlay:CLEar" on page 163
- ":CDISplay" on page 95

:DISPlay:SOURce

N (see page 530)

Command Syntax

```
:DISPlay:SOURce <value>
```

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

```
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 161
- ":DISPlay:DATA" on page 164

:DISPlay:VECTors

N (see page 530)

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 161

:EXTernal Trigger Commands

Control the input characteristics of the external trigger input. See "Introduction to :EXTernal Trigger Commands" on page 171.

Table 46 :EXTernal Trigger Commands Summary

Command	Query	Options and Query Returns
:EXTernal:BWLimit <bwlimit> (see page 173)	:EXTernal:BWLimit? (see page 173)	<bwlimit> ::= {0 | OFF}
:EXTernal:IMPedance <value> (see page 174)	:EXTernal:IMPedance? (see page 174)	<impedance> ::= {ONEMeg | FIFTy}
:EXTernal:PROBe <attenuation> (see page 175)	:EXTernal:PROBe? (see page 175)	<attenuation> ::= probe attenuation ratio in NR3 format
n/a	:EXTernal:PROBe:ID? (see page 176)	<probe id> ::= unquoted ASCII string up to 11 characters
:EXTernal:PROBe:STYPe <signal type> (see page 177)	:EXTernal:PROBe:STYPe? (see page 177)	<signal type> ::= {DIFFerential | SINGle}
:EXTernal:PROTection[:CLEar] (see page 178)	:EXTernal:PROTection? (see page 178)	{NORM | TRIP}
:EXTernal:RANGe <range>[<suffix>] (see page 179)	:EXTernal:RANGe? (see page 179)	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V | mV}
:EXTernal:UNITs <units> (see page 180)	:EXTernal:UNITs? (see page 180)	<units> ::= {VOLT | AMPere}

Introduction to :EXTernal Trigger Commands

The EXTernal trigger subsystem commands control the input characteristics of the external trigger input. The probe factor, impedance, input range, input protection state, units, and bandwidth limit settings may all be queried. Depending on the instrument type, some settings may be changeable.

Reporting the Setup

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

Return Format

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

```
:EXT:BWL 0;IMP ONEM;RANG +8.0E+00;UNIT VOLT;PROB +1.0E+00;PROB:STYP SING
```

:EXTernal:BWLimit

C (see page 530)

Command Syntax

```
:EXTernal:BWLimit <bwlimit>

<bwlimit> ::= {0 | OFF}
```

The :EXTernal:BWLimit command is provided for product compatibility. The only legal value is 0 or OFF. Use the :TRIGger:HFReject command to limit bandwidth on the external trigger input.

Query Syntax

```
:EXTernal:BWLimit?
```

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

Return Format

```
<bwlimit><NL>

<bwlimit> ::= 0
```

See Also

- "Introduction to :EXTernal Trigger Commands" on page 171
- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:HFReject" on page 317

:EXTernal:IMPedance

C (see page 530)

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).

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 171
- "Introduction to :TRIGger Commands" on page 313
- ":CHANnel<n>:IMPedance" on page 148

:EXTernal:PROBe

C (see page 530)

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 171
- ":EXTernal:RANGe" on page 179
- "Introduction to :TRIGger Commands" on page 313
- ":CHANnel<n>:PROBe" on page 152

:EXTernal:PROBe:ID

C (see page 530)

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 171

:EXTernal:PROBe:STYPe

C (see page 530)

Command Syntax

NOTE 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 171

:EXTernal:PROTection

N (see page 530)

Command Syntax

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

When the external trigger input impedance is set to 50Ω, 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 174) 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 171
- ":EXTernal:IMPedance" on page 174
- ":EXTernal:PROBe" on page 175

:EXTernal:RANGe

C (see page 530)

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 171
- ":EXTernal:PROBe" on page 175
- "Introduction to :TRIGger Commands" on page 313

:EXTernal:UNITs

N (see page 530)

Command Syntax

```
:EXTernal:UNITs <units>
```

```
<units> ::= {VOLT | AMPere}
```

The :EXTernal:UNITs command sets the measurement units for the probe connected to the external trigger input. Select VOLT for a voltage probe and select AMPere for a current probe. Measurement results, channel sensitivity, and trigger level will reflect the measurement units you select.

Query Syntax

```
:EXTernal:UNITs?
```

The :CHANnel<n>:UNITs? query returns the current units setting for the external trigger.

Return Format

```
<units><NL>
```

```
<units> ::= {VOLT | AMP}
```

See Also

- "Introduction to :EXTernal Trigger Commands" on page 171
- "Introduction to :TRIGger Commands" on page 313
- ":EXTernal:RANGe" on page 179
- ":EXTernal:PROBe" on page 175
- ":CHANnel<n>:UNITs" on page 159

:FUNCtion Commands

Control functions in the measurement/storage module. See "Introduction to :FUNCtion Commands" on page 182.

Table 47 :FUNCtion Commands Summary

Command	Query	Options and Query Returns
:FUNCtion:CENTer <frequency> (see page 183)	:FUNCtion:CENTer? (see page 183)	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 25 GHz.
:FUNCtion:DISPlay {{0 | OFF} | {1 | ON}} (see page 184)	:FUNCtion:DISPlay? (see page 184)	{0 | 1}
:FUNCtion:OFFSet <offset> (see page 185)	:FUNCtion:OFFSet? (see page 185)	<offset> ::= the value at center screen in NR3 format. The range of legal values is +/-10 times the current sensitivity of the selected function.
:FUNCtion:OPERation <operation> (see page 186)	:FUNCtion:OPERation? (see page 186)	<operation> ::= {SUBTract | MULTiply | INTegrate | DIFFerentiate | FFT | SQRT}
:FUNCtion:RANGe <range> (see page 187)	:FUNCtion:RANGe? (see page 187)	<range> ::= the full-scale vertical axis value in NR3 format. The range for ADD, SUBT, MULT is 8E-6 to 800E+3. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFFerentiate function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV.
:FUNCtion:REFerence <level> (see page 188)	:FUNCtion:REFerence? (see page 188)	<level> ::= 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).
:FUNCtion:SCALe <scale value>[<suffix>] (see page 189)	:FUNCtion:SCALe? (see page 189)	<scale value> ::= integer in NR1 format <suffix> ::= {V | dB}

Table 47 :FUNCtion Commands Summary (continued)

Command	**Query**	**Options and Query Returns**

`:FUNCtion:SOURce <source>` (see page 190)	`:FUNCtion:SOURce?` (see page 190)	`<source> ::= {CHANnel<n> | ADD | SUBT | MULT}`
		`<n> ::= 1-2 or 1-4 in NR1 format`

`:FUNCtion:SPAN <span>` (see page 191)	`:FUNCtion:SPAN?` (see page 191)	`<span> ::= the current frequency span in NR3 format.`
		`Legal values are 1 Hz to 100 GHz.`

`:FUNCtion:WINDow <window>` (see page 192)	`:FUNCtion:WINDow?` (see page 192)	`<window> ::= {RECTangular | HANNing | FLATtop}`

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.

```
:FUNC:OPER SUBT;DISP 0;RANG +8.00E+00;OFFS +0.00000E+00
```

:FUNCtion:CENTer

N (see page 530)

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 :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.

See Also

- "Introduction to :FUNCtion Commands" on page 182
- ":FUNCtion:SPAN" on page 191
- ":TIMebase:RANGe" on page 306
- ":TIMebase:SCALe" on page 308

:FUNCtion:DISPlay

N (see page 530)

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 182
- ":VIEW" on page 122
- ":BLANk" on page 94
- ":STATus" on page 119

:FUNCtion:OFFSet

N (see page 530)

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:REFerence 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 182
- ":FUNCtion:RANGe" on page 187
- ":FUNCtion:REFerence" on page 188
- ":FUNCtion:SCALe" on page 189

:FUNCtion:OPERation

N (see page 530)

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 182

:FUNCtion:RANGe

N (see page 530)

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 182
- ":FUNCtion:SCALe" on page 189

:FUNCtion:REFerence

N (see page 530)

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 182
- ":FUNCtion:OFFSet" on page 185
- ":FUNCtion:RANGe" on page 187
- ":FUNCtion:SCALe" on page 189

:FUNCtion:SCALe

N (see page 530)

Command Syntax

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

```
<scale value> ::= integer in NR1 format
```

```
<suffix> ::= {V | 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 182
- ":FUNCtion:RANGe" on page 187

:FUNCtion:SOURce

N (see page 530)

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 182
- ":FUNCtion:OPERation" on page 186

:FUNCtion:SPAN

N (see page 530)

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 a *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 182
- ":FUNCtion:CENTer" on page 183
- ":TIMebase:RANGe" on page 306
- ":TIMebase:SCALe" on page 308

:FUNCtion:WINDow

N (see page 530)

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 182

:HARDcopy Commands

Set and query the selection of hardcopy device and formatting options. See "Introduction to :HARDcopy Commands" on page 193.

Table 48 :HARDcopy Commands Summary

Command	Query	Options and Query Returns
`:HARDcopy:AREA <area>` (see page 195)	`:HARDcopy:AREA?` (see page 195)	`<area> ::= SCReen`

`:HARDcopy:APRinter <active_printer>` (see page 196)	`:HARDcopy:APRinter?` (see page 196)	`<active_printer> ::= {<index> | <name>}`
		`<index> ::= integer index of printer in list`
		`<name> ::= name of printer in list`

`:HARDcopy:FACTors {{0 | OFF} | {1 | ON}}` (see page 197)	`:HARDcopy:FACTors?` (see page 197)	`{0 | 1}`
`:HARDcopy:FFEed {{0 | OFF} | {1 | ON}}` (see page 198)	`:HARDcopy:FFEed?` (see page 198)	`{0 | 1}`
`:HARDcopy:INKSaver {{0 | OFF} | {1 | ON}}` (see page 199)	`:HARDcopy:INKSaver?` (see page 199)	`{0 | 1}`
`:HARDcopy:PALette <palette>` (see page 200)	`:HARDcopy:PALette?` (see page 200)	`<palette> ::= {COLor | GRAYscale | NONE}`

n/a	`:HARDcopy:PRinter:LIST?` (see page 201)	`<list> ::= [<printer_spec>] ... [printer_spec>]`
		`<printer_spec> ::= "<index>,<active>,<name>;"`
		`<index> ::= integer index of printer`
		`<active> ::= {Y | N}`
		`<name> ::= name of printer`

`:HARDcopy:STARt` (see page 202)	n/a	n/a

Introduction to :HARDcopy Commands

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

:HARDC is an acceptable short form for :HARDcopy.

Reporting the Setup

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

Return Format

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

```
:HARD:APR "";AREA SCR;FACT 0;FFE 0;INKS 0;PAL NONE
```

:HARDcopy:AREA

N (see page 530)

Command Syntax

```
:HARDcopy:AREA <area>

<area> ::= SCReen
```

The :HARDcopy:AREA command controls what part of the display area is printed. Currently, the only legal choice is SCReen.

Query Syntax

```
:HARDcopy:AREA?
```

The :HARDcopy:AREA? query returns the selected display area.

Return Format

```
<area><NL>

<area> ::= SCR
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:STARt" on page 202
- ":HARDcopy:APRinter" on page 196
- ":HARDcopy:PRinter:LIST" on page 201
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:INKSaver" on page 199
- ":HARDcopy:PALette" on page 200

:HARDcopy:APRinter

N (see page 530)

Command Syntax

```
:HARDcopy:APRinter <active_printer>
```

```
<active_printer> ::= {<index> | <name>}
```

```
<index> ::= integer index of printer in list
```

```
<name> ::= name of printer in list
```

The :HARDcopy:APRinter command sets the active printer.

Query Syntax

```
:HARDcopy:APRinter?
```

The :HARDcopy:APRinter? query returns the name of the active printer.

Return Format

```
<name><NL>
```

```
<name> ::= name of printer in list
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:PRinter:LIST" on page 201
- ":HARDcopy:STARt" on page 202

:HARDcopy:FACTors

N (see page 530)

Command Syntax

```
:HARDcopy:FACTors <factors>
```

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

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

Query Syntax

```
:HARDcopy:FACTors?
```

The :HARDcopy:FACTors? query returns a flag indicating whether oscilloscope instrument settings are output on the hardcopy.

Return Format

```
<factors><NL>
```

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

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:STARt" on page 202
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:INKSaver" on page 199
- ":HARDcopy:PALette" on page 200

:HARDcopy:FFEed

N (see page 530)

Command Syntax

```
:HARDcopy:FFEed <ffeed>
```

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

The HARDcopy:FFEed command controls whether a formfeed is output between the screen image and factors of a hardcopy dump.

ON (or 1) is only valid when PRINter0 or PRINter1 is set as the :HARDcopy:FORMat type.

Query Syntax

```
:HARDcopy:FFEed?
```

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

Return Format

```
<ffeed><NL>
```

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

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:STARt" on page 202
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:INKSaver" on page 199
- ":HARDcopy:PALette" on page 200

:HARDcopy:INKSaver

N (see page 530)

Command Syntax

```
:HARDcopy:INKSaver <value>

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

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

Query Syntax

```
:HARDcopy:INKSaver?
```

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:STARt" on page 202
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:PALette" on page 200

:HARDcopy:PALette

N (see page 530)

Command Syntax

```
:HARDcopy:PALette <palette>

<palette> ::= {COLor | GRAYscale | NONE}
```

The HARDcopy:PALette command sets the hardcopy palette color.

NOTE If no printer is connected, NONE is the only valid parameter.

Query Syntax

```
:HARDcopy:PALette?
```

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

Return Format

```
<palette><NL>

<palette> ::= {COL | GRAY | NONE}
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:STARt" on page 202
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:INKSaver" on page 199

:HARDcopy:PRinter:LIST

N (see page 530)

Query Syntax

```
:HARDcopy:PRinter:LIST?
```

The :HARDcopy:PRinter:LIST? query returns a list of available printers. The list can be empty.

Return Format

```
<list><NL>

<list> ::= [<printer_spec>] ... [printer_spec>]

<printer_spec> ::= "<index>,<active>,<name>;"

<index> ::= integer index of printer

<active> ::= {Y | N}

<name> ::= name of printer (for example "DESKJET 950C")
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:APRinter" on page 196
- ":HARDcopy:STARt" on page 202

:HARDcopy:STARt

N (see page 530)

Command Syntax

```
:HARDcopy:STARt
```

The :HARDcopy:STARt command starts a print job.

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:APRinter" on page 196
- ":HARDcopy:PRinter:LIST" on page 201
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:INKSaver" on page 199
- ":HARDcopy:PALette" on page 200

: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 204.

Table 49 :MARKer Commands Summary

Command	Query	Options and Query Returns
:MARKer:MODE <mode> (see page 205)	:MARKer:MODE? (see page 205)	<mode> ::= {OFF | MEASurement | MANual}
:MARKer:X1Position <position>[suffix] (see page 206)	:MARKer:X1Position? (see page 206)	<position> ::= X1 cursor position value in NR3 format [suffix] ::= {s | ms | us | ns | ps | Hz | kHz | MHz} <return_value> ::= X1 cursor position value in NR3 format
:MARKer:X1Y1source <source> (see page 207)	:MARKer:X1Y1source? (see page 207)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>
:MARKer:X2Position <position>[suffix] (see page 208)	:MARKer:X2Position? (see page 208)	<position> ::= X2 cursor position value in NR3 format [suffix] ::= {s | ms | us | ns | ps | Hz | kHz | MHz} <return_value> ::= X2 cursor position value in NR3 format
:MARKer:X2Y2source <source> (see page 209)	:MARKer:X2Y2source? (see page 209)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>
n/a	:MARKer:XDELta? (see page 210)	<return_value> ::= X cursors delta value in NR3 format
:MARKer:Y1Position <position>[suffix] (see page 211)	:MARKer:Y1Position? (see page 211)	<position> ::= Y1 cursor position value in NR3 format [suffix] ::= {V | mV | dB} <return_value> ::= Y1 cursor position value in NR3 format

Table 49 :MARKer Commands Summary (continued)

Command	Query	Options and Query Returns

`:MARKer:Y2Position <position>[suffix]` (see page 212)	`:MARKer:Y2Position?` (see page 212)	`<position> ::= Y2 cursor position value in NR3 format`
		`[suffix] ::= {V | mV | dB}`
		`<return_value> ::= Y2 cursor position value in NR3 format`

n/a	`:MARKer:YDELta?` (see page 213)	`<return_value> ::= Y cursors delta value in NR3 format`

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
```

:MARKer:MODE

N (see page 530)

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 204
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MEASure:SOURce" on page 240

:MARKer:X1Position

N (see page 530)

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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X2Position" on page 208
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MEASure:TSTArt" on page 484

:MARKer:X1Y1source

N (see page 530)

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 205).

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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X2Y2source" on page 209
- ":MEASure:SOURce" on page 240

:MARKer:X2Position

N (see page 530)

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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Position" on page 206
- ":MARKer:X2Y2source" on page 209
- ":MEASure:TSTOp" on page 485

:MARKer:X2Y2source

N (see page 530)

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 205).

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

```
<source><NL>

<source> ::= {CHAN<n> | FUNC | NONE}
```

See Also

- "Introduction to :MARKer Commands" on page 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Y1source" on page 207
- ":MEASure:SOURce" on page 240

:MARKer:XDELta

N (see page 530)

Query Syntax

```
:MARKer:XDELta?
```

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

Xdelta = (Value at X2 cursor) - (Value at X1 cursor)

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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Position" on page 206
- ":MARKer:X2Position" on page 208
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209

:MARKer:Y1Position

N (see page 530)

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.

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

Return Format

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

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

See Also

- "Introduction to :MARKer Commands" on page 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MARKer:Y2Position" on page 212
- ":MEASure:VSTArt" on page 490

:MARKer:Y2Position

N (see page 530)

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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MARKer:Y1Position" on page 211
- ":MEASure:VSTOp" on page 491

:MARKer:YDELta

N (see page 530)

Query Syntax

```
:MARKer:YDELta?
```

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

Ydelta = (Value at Y2 cursor) - (Value at Y1 cursor)

NOTE If the front-panel cursors are off or are set to Binary or Hex Mode, the marker position values are not defined. Make sure to set :MARKer:MODE to MANual 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 204
- ":MARKer:MODE" on page 205
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MARKer:Y1Position" on page 211
- ":MARKer:Y2Position" on page 212

:MEASure Commands

Select automatic measurements to be made and control time markers. See "Introduction to :MEASure Commands" on page 218.

Table 50 :MEASure Commands Summary

Command	Query	Options and Query Returns
:MEASure:CLEar (see page 220)	n/a	n/a
:MEASure:COUNter [<source>] (see page 221)	:MEASure:COUNter? [<source>] (see page 221)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= counter frequency in Hertz in NR3 format
:MEASure:DEFine DELay, <delay spec> (see page 222)	:MEASure:DEFine? DELay (see page 223)	<delay spec> ::= <edge_spec1>,<edge_spec2> edge_spec1 ::= [<slope>]<occurrence> edge_spec2 ::= [<slope>]<occurrence> <slope> ::= {+ | -} <occurrence> ::= integer

:MEASure:DEFine THResholds, <threshold spec> (see page 222)	:MEASure:DEFine? THResholds (see page 223)	<threshold spec> ::= {STANdard} | {<threshold mode>,<upper>, <middle>,<lower>}
		<threshold mode> ::= {PERCent | ABSolute}

:MEASure:DELay [<source1>] [,<source2>] (see page 225)	:MEASure:DELay? [<source1>] [,<source2>] (see page 225)	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUTYcycle [<source>] (see page 227)	:MEASure:DUTYcycle? [<source>] (see page 227)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format
:MEASure:FALLtime [<source>] (see page 228)	:MEASure:FALLtime? [<source>] (see page 228)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds between the lower and upper thresholds in NR3 format

Table 50 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:FREQuency [<source>] (see page 229)	:MEASure:FREQuency? [<source>] (see page 229)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= frequency in Hertz in NR3 format
:MEASure:NWIDth [<source>] (see page 230)	:MEASure:NWIDth? [<source>] (see page 230)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>] (see page 231)	:MEASure:OVERshoot? [<source>] (see page 231)	<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
:MEASure:PERiod [<source>] (see page 233)	:MEASure:PERiod? [<source>] (see page 233)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source1>] [,<source2>] (see page 234)	:MEASure:PHASe? [<source1>] [,<source2>] (see page 234)	<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
:MEASure:PREShoot [<source>] (see page 235)	:MEASure:PREShoot? [<source>] (see page 235)	<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
:MEASure:PWIDth [<source>] (see page 236)	:MEASure:PWIDth? [<source>] (see page 236)	<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

Table 50 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:RISEtime [<source>] (see page 237)	:MEASure:RISEtime? [<source>] (see page 237)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= rise time in seconds in NR3 format
:MEASure:SDEViation [<source>] (see page 238)	:MEASure:SDEViation? [<source>] (see page 238)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated std deviation in NR3 format
:MEASure:SHOW {1 | ON} (see page 239)	:MEASure:SHOW? (see page 239)	{1}

:MEASure:SOURce [<source1>] [,<source2>] (see page 240)	:MEASure:SOURce? (see page 240)	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH}
		<n> ::= 1-2 or 1-4 in NR1 format
		<return_value> ::= {<source> | NONE}

n/a	:MEASure:TEDGe? <slope><occurrence>[, <source>] (see page 242)	<slope> ::= direction of the waveform <occurrence> ::= the transition to be reported <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds of the specified transition
n/a	:MEASure:TVALue? <value>, [<slope>]<occurrence> [,<source>] (see page 244)	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <return_value> ::= time in seconds of specified voltage crossing in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format

Table 50 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
:MEASure:VAMPlitude [<source>] (see page 246)	:MEASure:VAMPlitude? [<source>] (see page 246)	<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 247)	:MEASure:VAVerage? [<source>] (see page 247)	<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 248)	:MEASure:VBASe? [<source>] (see page 248)	<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 249)	:MEASure:VMAX? [<source>] (see page 249)	<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 250)	:MEASure:VMIN? [<source>] (see page 250)	<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 251)	:MEASure:VPP? [<source>] (see page 251)	<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
:MEASure:VRMS [<source>] (see page 252)	:MEASure:VRMS? [<source>] (see page 252)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated dc RMS voltage in NR3 format

Table 50 :MEASure Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:MEASure:VTIMe? <vtime>[,<source>] (see page 253)	<vtime> ::= displayed time from trigger in seconds in NR3 format <return_value> ::= voltage at the specified time in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:MEASure:VTOP [<source>] (see page 254)	:MEASure:VTOP? [<source>] (see page 254)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= voltage at the top of the waveform in NR3 format
:MEASure:XMAX [<source>] (see page 255)	:MEASure:XMAX? [<source>] (see page 255)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format
:MEASure:XMIN [<source>] (see page 256)	:MEASure:XMIN? [<source>] (see page 256)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format

Introduction to :MEASure Commands

The commands in the MEASure subsystem are used to make parametric measurements on displayed waveforms.

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the oscilloscope screen.

Measurement Type	Portion of waveform that must be displayed
period, duty cycle, or frequency	at least one complete cycle
pulse width	the entire pulse
rise time	rising edge, top and bottom of pulse
fall time	falling edge, top and bottom of pulse

Measurement Error

If a measurement cannot be made (typically because the proper portion of the waveform is not displayed), the value +9.9E+37 is returned for that measurement.

Making Measurements

If more than one waveform, edge, or pulse is displayed, time measurements are made on the portion of the displayed waveform closest to the trigger reference (left, center, or right).

When making measurements in the 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 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
```

:MEASure:CLEar

N (see page 530)

Command Syntax

```
:MEASure:CLEar
```

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

See Also

- "Introduction to :MEASure Commands" on page 218

:MEASure:COUNter

N (see page 530)

Command Syntax

```
:MEASure:COUNter [<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 :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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:FREQuency" on page 229
- ":MEASure:CLEar" on page 220

:MEASure:DEFine

N (see page 530)

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.

MEASure Command	DELay	THResholds
DUTYcycle		x
DELay	x	x
FALLtime		x
FREQuency		x
NWIDth		x
OVERshoot		x
PERiod		x
PHASe		x
PREShoot		x
PWIDth		x
RISetime		x
VAVerage		x
VRMS		x

: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:

delay = t(<edge_spec2>) - t(<edge_spec1>)

NOTE

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

:MEASure:DEFine THResholds Command Syntax

```
:MEASure:DEFine THResholds,<threshold spec>
```

```
<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 158: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.

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 218
- ":MEASure:DELay" on page 225
- ":MEASure:SOURce" on page 240
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:SCALe" on page 158
- ":CHANnel<n>:PROBe" on page 152
- ":CHANnel<n>:UNITs" on page 159

:MEASure:DELay

N (see page 530)

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:

delay = t(<edge spec 2>) - t(<edge spec 1>)

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

NOTE

The :MEASure:DELay command and the front-panel delay measurement differ from the :MEASure:DELay? query.

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

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

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

Query Syntax

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

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

Also in the :MEASure:DEFine command, you can set upper, middle, and lower threshold values. *It is the middle threshold value that is used when performing the delay query.* The standard upper, middle, and lower measurement thresholds are 90%, 50%, and 10% values between Vbase and

Vtop. If you want to move the delay measurement point nearer to Vtop or Vbase, you must change the threshold values with the :MEASure:DEFine THResholds command.

Return Format

```
<value><NL>

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

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:DEFine" on page 222
- ":MEASure:PHASe" on page 234

:MEASure:DUTYcycle

C (see page 530)

Command Syntax

```
:MEASure:DUTYcycle [<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: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 cycle = (+pulse width/period)*100

Return Format

```
<value><NL>

<value> ::= ratio of positive pulse width to period in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:PERiod" on page 233
- ":MEASure:PWIDth" on page 236
- ":MEASure:SOURce" on page 240

Example Code

- "Example Code" on page 240

:MEASure:FALLtime

C (see page 530)

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:

fall time = time at lower threshold - 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 218
- ":MEASure:RISetime" on page 237
- ":MEASure:SOURce" on page 240

:MEASure:FREQuency

C (see page 530)

Command Syntax

```
:MEASure:FREQuency [<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:FREQuency command installs a screen measurement and starts a frequency measurement. If the optional source parameter is specified, the current source is modified.

IF the edge on the screen closest to the trigger reference is rising:

THEN frequency = 1/(time at trailing rising edge - time at leading rising edge)

ELSE frequency = 1/(time at trailing falling edge - time at leading falling edge)

NOTE This command is not available if the source is FFT (Fast Fourier Transform).

Query Syntax

```
:MEASure:FREQuency? [<source>]
```

The :MEASure:FREQuency? query measures and outputs the frequency of the cycle on the screen closest to the trigger reference.

Return Format

```
<source><NL>

<source> ::= frequency in Hertz in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:SOURce" on page 240
- ":MEASure:PERiod" on page 233

Example Code

- "Example Code" on page 240

:MEASure:NWIDth

🟩C (see page 530)

Command Syntax

```
:MEASure:NWIDth [<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: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:

width = (time at trailing rising edge - time at leading falling edge)

Return Format

```
<value><NL>

<value> ::= negative pulse width in seconds in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:SOURce" on page 240
- ":MEASure:PWIDth" on page 236
- ":MEASure:PERiod" on page 233

:MEASure:OVERshoot

C (see page 530)

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 218
- ":MEASure:PREShoot" on page 235
- ":MEASure:SOURce" on page 240
- ":MEASure:VMAX" on page 249

- ":MEASure:VTOP" on page 254
- ":MEASure:VBASe" on page 248
- ":MEASure:VMIN" on page 250

:MEASure:PERiod

C (see page 530)

Command Syntax

```
:MEASure:PERiod [<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: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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:NWIDth" on page 230
- ":MEASure:PWIDth" on page 236
- ":MEASure:FREQuency" on page 229

Example Code

- "Example Code" on page 240

:MEASure:PHASe

N (see page 530)

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:

phase = (delay / period of input 1) x 360

Return Format

```
<value><NL>

<value> ::= the phase angle value in degrees in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:DELay" on page 225
- ":MEASure:PERiod" on page 233
- ":MEASure:SOURce" on page 240

:MEASure:PREShoot

C (see page 530)

Command Syntax

```
: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

```
: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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VMIN" on page 250
- ":MEASure:VMAX" on page 249
- ":MEASure:VTOP" on page 254
- ":MEASure:VBASe" on page 248

:MEASure:PWIDth

C (see page 530)

Command Syntax

```
:MEASure:PWIDth [<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: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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:NWIDth" on page 230
- ":MEASure:PERiod" on page 233

:MEASure:RISetime

C (see page 530)

Command Syntax

```
: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

```
: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:

rise time = time at upper threshold - time at lower threshold

Return Format

```
<value><NL>

<value> ::= rise time in seconds in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:SOURce" on page 240
- ":MEASure:FALLtime" on page 228

:MEASure:SDEViation

N (see page 530)

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 218
- ":MEASure:SOURce" on page 240

:MEASure:SHOW

N (see page 530)

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 218

:MEASure:SOURce

C (see page 530)

Command Syntax

```
:MEASure:SOURce <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: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> ::= {CHAN<n> | FUNC | NONE}
```

See Also:

- "Introduction to :MEASure Commands" on page 218
- ":MARKer:MODE" on page 205
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Y2source" on page 209
- ":MEASure:DELay" on page 225
- ":MEASure:PHASe" on page 234

Example Code

```
' MEASURE - The commands in the MEASURE subsystem are used to make
' measurements on displayed waveforms.
myScope.WriteString ":MEASURE:SOURCE CHANNEL1"    ' Source to measure.
myScope.WriteString ":MEASURE:FREQUENCY?"    ' Query for frequency.
varQueryResult = myScope.ReadNumber    ' Read frequency.
MsgBox "Frequency:" + vbCrLf _
      + FormatNumber(varQueryResult / 1000, 4) + " kHz"
myScope.WriteString ":MEASURE:DUTYCYCLE?"    ' Query for duty cycle.
```

```
   varQueryResult = myScope.ReadNumber    ' Read duty cycle.
   MsgBox "Duty cycle:" + vbCrLf _
         + FormatNumber(varQueryResult, 3) + "%"
   myScope.WriteString ":MEASURE:RISETIME?"    ' Query for risetime.
   varQueryResult = myScope.ReadNumber    ' Read risetime.
   MsgBox "Risetime:" + vbCrLf _
         + FormatNumber(varQueryResult * 1000000, 4) + " us"
   myScope.WriteString ":MEASURE:VPP?"    ' Query for Pk to Pk voltage.
   varQueryResult = myScope.ReadNumber    ' Read VPP.
   MsgBox "Peak to peak voltage:" + vbCrLf _
         + FormatNumber(varQueryResult, 4) + " V"
   myScope.WriteString ":MEASURE:VMAX?"    ' Query for Vmax.
   varQueryResult = myScope.ReadNumber    ' Read Vmax.
   MsgBox "Maximum voltage:" + vbCrLf _
         + FormatNumber(varQueryResult, 4) + " V"
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:MEASure:TEDGe

N (see page 530)

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> ::= {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:TEDGe query is sent, the displayed signal is searched for the specified transition. The time interval between the trigger event and this occurrence is returned as the response to the query. The sign of the slope selects a rising (+) or falling (-) edge. If no sign is specified for the slope, it is assumed to be the rising edge.

The magnitude of occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the midpoint threshold in the positive direction. Once this crossing is found, the oscilloscope reports the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope reports +9.9E+37. This value is returned if the waveform does not cross the specified vertical value, or if the waveform does not cross the specified vertical value for the specific number of times in the direction specified.

You can make delay and phase measurements using the MEASure:TEDGe command:

Delay = time at the nth rising or falling edge of the channel - time at the same edge of another channel

Phase = (delay between channels / period of channel) x 360

For an example of making a delay and phase measurement, see ":MEASure:TEDGe Code" on page 243.

If the optional source parameter is specified, the current source is modified.

NOTE This query is not available if the source is FFT (Fast Fourier Transform).

Return Format

```
<value><NL>

<value> ::= time in seconds of the specified transition in NR3 format
```

:MEASure:TEDGe Code

```
' 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 578

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:TVALue" on page 244
- ":MEASure:VTIMe" on page 253

:MEASure:TVALue

C (see page 530)

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 218
- ":MEASure:TEDGe" on page 242
- ":MEASure:VTIMe" on page 253

:MEASure:VAMPlitude

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VBASe" on page 248
- ":MEASure:VTOP" on page 254
- ":MEASure:VPP" on page 251

:MEASure:VAVerage

C (see page 530)

Command Syntax

```
:MEASure:VAVerage [<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: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 218
- ":MEASure:SOURce" on page 240

:MEASure:VBASe

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VTOP" on page 254
- ":MEASure:VAMPlitude" on page 246
- ":MEASure:VMIN" on page 250

:MEASure:VMAX

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VMIN" on page 250
- ":MEASure:VPP" on page 251
- ":MEASure:VTOP" on page 254

:MEASure:VMIN

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VBASe" on page 248
- ":MEASure:VMAX" on page 249
- ":MEASure:VPP" on page 251

:MEASure:VPP

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VMAX" on page 249
- ":MEASure:VMIN" on page 250
- ":MEASure:VAMPlitude" on page 246

:MEASure:VRMS

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240

:MEASure:VTIMe

N (see page 530)

Query Syntax

```
:MEASure:VTIMe? <vtime_argument>[,<source>]
```

```
<vtime_argument> ::= time from trigger in seconds
```

```
<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: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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:TEDGe" on page 242
- ":MEASure:TVALue" on page 244

:MEASure:VTOP

C (see page 530)

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 218
- ":MEASure:SOURce" on page 240
- ":MEASure:VMAX" on page 249
- ":MEASure:VAMPlitude" on page 246
- ":MEASure:VBASe" on page 248

:MEASure:XMAX

N (see page 530)

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 218
- ":MEASure:XMIN" on page 256
- ":MEASure:TMAX" on page 482

:MEASure:XMIN

N (see page 530)

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 218
- ":MEASure:XMAX" on page 255
- ":MEASure:TMIN" on page 483

:RECall Commands

Recall previously saved oscilloscope setups and traces. See "Introduction to :RECall Commands" on page 257.

Table 51 :RECall Commands Summary

Command	Query	Options and Query Returns
`:RECall:FILename <base_name>` (see page 258)	`:RECall:FILename?` (see page 258)	`<base_name> ::= quoted ASCII string`

`:RECall:IMAGe[:STARt] [<file_spec>]` (see page 259)	n/a	`<file_spec> ::= {<internal_loc> | <file_name>}`
		`<internal_loc> ::= 0-9; an integer in NR1 format`
		`<file_name> ::= quoted ASCII string`

n/a	`:RECall:PWD?` (see page 260)	`<path_info> ::= quoted ASCII string`

`:RECall:SETup[:STARt] [<file_spec>]` (see page 261)	n/a	`<file_spec> ::= {<internal_loc> | <file_name>}`
		`<internal_loc> ::= 0-9; an integer in NR1 format`
		`<file_name> ::= quoted ASCII string`

Introduction to :RECall Commands

The :RECall subsystem provides commands to recall previously saved oscilloscope setups and traces.

Reporting the Setup

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

Return Format

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

```
:REC:FIL "scope_0"
```

:RECall:FILename

N (see page 530)

Command Syntax

```
:RECall:FILename <base_name>
```

```
<base_name> ::= quoted ASCII string
```

The :RECall:FILename command specifies the source for any RECall operations.

NOTE This command specifies a file's base name only, without path information or an extension.

Query Syntax

```
:RECall:FILename?
```

The :RECall:FILename? query returns the current RECall filename.

Return Format

```
<base_name><NL>
```

```
<base_name> ::= quoted ASCII string
```

See Also

- "Introduction to :RECall Commands" on page 257
- ":RECall:IMAGe[:STARt]" on page 259
- ":RECall:SETup[:STARt]" on page 261
- ":SAVE:FILename" on page 264

:RECall:IMAGe[:STARt]

N (see page 530)

Command Syntax

```
:RECall:IMAGe[:STARt] [<file_spec>]

<file_spec> ::= {<internal_loc> | <file_name>}

<internal_loc> ::= 0-9; an integer in NR1 format

<file_name> ::= quoted ASCII string
```

The :RECall:IMAGe[:STARt] command recalls a trace (TIFF) image.

NOTE If a file extension is provided as part of a specified <file_name>, it must be ".tif".

See Also

- "Introduction to :RECall Commands" on page 257
- ":RECall:FILename" on page 258
- ":SAVE:IMAGe[:STARt]" on page 265

:RECall:PWD

N (see page 530)

Query Syntax

```
:RECall:PWD?
```

The :RECall:PWD? query returns the current recall path information.

Return Format

```
<path_info><NL>

<path_info> ::= quoted ASCII string
```

See Also

- "Introduction to :RECall Commands" on page 257
- ":SAVE:PWD" on page 271

:RECall:SETup[:STARt]

N (see page 530)

Command Syntax

```
:RECall:SETup[:STARt] [<file_spec>]

<file_spec> ::= {<internal_loc> | <file_name>}

<internal_loc> ::= 0-9; an integer in NR1 format

<file_name> ::= quoted ASCII string
```

The :RECall:SETup[:STARt] command recalls an oscilloscope setup.

NOTE If a file extension is provided as part of a specified <file_name>, it must be ".scp".

See Also

- "Introduction to :RECall Commands" on page 257
- ":RECall:FILename" on page 258
- ":SAVE:SETup[:STARt]" on page 272

:SAVE Commands

Save oscilloscope setups and traces, screen images, and data. See "Introduction to :SAVE Commands" on page 263.

Table 52 :SAVE Commands Summary

Command	Query	Options and Query Returns
:SAVE:FILename <base_name> (see page 264)	:SAVE:FILename? (see page 264)	<base_name> ::= quoted ASCII string
:SAVE:IMAGe[:STARt] [<file_spec>] (see page 265)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:IMAGe:AREA <area> (see page 266)	:SAVE:IMAGe:AREA? (see page 266)	<area> ::= {GRATicule | SCReen}
:SAVE:IMAGe:FACTors {{0 | OFF} | {1 | ON}} (see page 267)	:SAVE:IMAGe:FACTors? (see page 267)	{0 | 1}
:SAVE:IMAGe:FORMat <format> (see page 268)	:SAVE:IMAGe:FORMat? (see page 268)	<format> ::= {TIFF | {BMP | BMP24bit} | BMP8bit | PNG | NONE}
:SAVE:IMAGe:INKSaver {{0 | OFF} | {1 | ON}} (see page 269)	:SAVE:IMAGe:INKSaver? (see page 269)	{0 | 1}
:SAVE:IMAGe:PALette <palette> (see page 270)	:SAVE:IMAGe:PALette? (see page 270)	<palette> ::= {COLor | GRAYscale | MONochrome}
n/a	:SAVE:PWD? (see page 271)	<path_info> ::= quoted ASCII string
:SAVE:SETup[:STARt] [<file_spec>] (see page 272)	n/a	<file_spec> ::= {<internal_loc> | <file_name>} <internal_loc> ::= 0-9; an integer in NR1 format <file_name> ::= quoted ASCII string
:SAVE:WAVeform[:STARt] [<file_name>] (see page 273)	n/a	<file_name> ::= quoted ASCII string

Table 52 :SAVE Commands Summary (continued)

Command	Query	Options and Query Returns
`:SAVE:WAVeform:FORMat <format>` (see page 274)	`:SAVE:WAVeform:FORMat?` (see page 274)	`<format> ::= {ALB | ASCiixy | CSV | BINary | NONE}`
`:SAVE:WAVeform:LENGth <length>` (see page 275)	`:SAVE:WAVeform:LENGth?` (see page 275)	`<length> ::= 100 to max. length; an integer in NR1 format`

Introduction to :SAVE Commands

The :SAVE subsystem provides commands to save oscilloscope setups and traces, screen images, and data.

:SAV is an acceptable short form for :SAVE.

Reporting the Setup

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

Return Format

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

```
:SAVE:FIL "scope_0";:SAVE:IMAG:AREA GRAT;FACT 0;FORM TIFF;INKS 0;
PAL MON;:SAVE:WAV:FORM NONE
```

:SAVE:FILename

N (see page 530)

Command Syntax

```
:SAVE:FILename <base_name>
```

```
<base_name> ::= quoted ASCII string
```

The :SAVE:FILename command specifies the source for any SAVE operations.

NOTE This command specifies a file's base name only, without path information or an extension.

Query Syntax

```
:SAVE:FILename?
```

The :SAVE:FILename? query returns the current SAVE filename.

Return Format

```
<base_name><NL>
```

```
<base_name> ::= quoted ASCII string
```

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:SETup[:STARt]" on page 272
- ":SAVE:WAVeform[:STARt]" on page 273
- ":SAVE:PWD" on page 271
- ":RECall:FILename" on page 258

:SAVE:IMAGe[:STARt]

N (see page 530)

Command Syntax

```
:SAVE:IMAGe[:STARt] [<file_spec>]

<file_spec> ::= {<internal_loc> | <file_name>}

<internal_loc> ::= 0-9; an integer in NR1 format

<file_name> ::= quoted ASCII string
```

The :SAVE:IMAGe[:STARt] command saves an image.

NOTE If a file extension is provided as part of a specified <file_name>, it must match the extension expected by the format specified in :SAVE:IMAGe:FORMat.

NOTE The <internal_loc> option is only valid if :SAVE:IMAGe:FORMat is TIFF.

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:INKSaver" on page 269
- ":SAVE:IMAGe:PALette" on page 270
- ":SAVE:FILename" on page 264
- ":RECall:IMAGe[:STARt]" on page 259

:SAVE:IMAGe:AREA

N (see page 530)

Command Syntax

```
:SAVE:IMAGe:AREA <area>
```

```
<area> ::= {GRATicule | SCReen}
```

The :SAVE:IMAGe:AREA command sets the area that will be saved as part of the image. If the :SAVE:IMAGe:FORMat is TIFF, the area is GRATicule. Otherwise, it is SCReen.

Query Syntax

```
:SAVE:IMAGe:AREA?
```

The :SAVE:IMAGe:AREA? query returns the selected image area.

Return Format

```
<area><NL>
```

```
<area> ::= {GRAT | SCR}
```

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:INKSaver" on page 269
- ":SAVE:IMAGe:PALette" on page 270

:SAVE:IMAGe:FACTors

N (see page 530)

Command Syntax

```
:SAVE:IMAGe:FACTors <factors>

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

The :SAVE:IMAGe:FACTors command controls whether the oscilloscope factors are output along with the image.

NOTE Factors are written to a separate file with the same path and base name but with the ".txt" extension.

Query Syntax

```
:SAVE:IMAGe:FACTors?
```

The :SAVE:IMAGe:FACTors? query returns a flag indicating whether oscilloscope factors are output along with the image.

Return Format

```
<factors><NL>

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

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:INKSaver" on page 269
- ":SAVE:IMAGe:PALette" on page 270

:SAVE:IMAGe:FORMat

N (see page 530)

Command Syntax

```
:SAVE:IMAGe:FORMat <format>

<format> ::= {TIFF | {BMP | BMP24bit} | BMP8bit | PNG}
```

The :SAVE:IMAGe:FORMat command sets the image format type.

Query Syntax

```
:SAVE:IMAGe:FORMat?
```

The :SAVE:IMAGe:FORMat? query returns the selected image format type.

Return Format

```
<format><NL>

<format> ::= {TIFF | BMP | BMP8 | PNG | NONE}
```

When NONE is returned, it indicates that a waveform data file format is currently selected.

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:INKSaver" on page 269
- ":SAVE:IMAGe:PALette" on page 270
- ":SAVE:WAVeform:FORMat" on page 274

:SAVE:IMAGe:INKSaver

N (see page 530)

Command Syntax

```
:SAVE:IMAGe:INKSaver <value>
```

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

The :SAVE:IMAGe:INKSaver command controls whether the graticule colors are inverted or not.

Query Syntax

```
:SAVE:IMAGe:INKSaver?
```

The :SAVE:IMAGe:INKSaver? query returns a flag indicating whether graticule colors are inverted or not.

Return Format

```
<value><NL>
```

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

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:PALette" on page 270

:SAVE:IMAGe:PALette

N (see page 530)

Command Syntax

```
:SAVE:IMAGe:PALette <palette>

<palette> ::= {COLor | GRAYscale | MONochrome}
```

The :SAVE:IMAGe:PALette command sets the image palette color.

NOTE MONochrome is the only valid choice when the :SAVE:IMAGe:FORMat is TIFF. COLor and GRAYscale are the only valid choices when the format is not TIFF.

Query Syntax

```
:SAVE:IMAGe:PALette?
```

The :SAVE:IMAGe:PALette? query returns the selected image palette color.

Return Format

```
<palette><NL>

<palette> ::= {COL | GRAY | MON}
```

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:INKSaver" on page 269

:SAVE:PWD

N (see page 530)

Query Syntax

```
:SAVE:PWD?
```

The :SAVE:PWD? query returns the current save path information.

Return Format

```
<path_info><NL>
```

```
<path_info> ::= quoted ASCII string
```

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:FILename" on page 264
- ":RECall:PWD" on page 260

:SAVE:SETup[:STARt]

N (see page 530)

Command Syntax

```
:SAVE:SETup[:STARt] [<file_spec>]

<file_spec> ::= {<internal_loc> | <file_name>}

<internal_loc> ::= 0-9; an integer in NR1 format

<file_name> ::= quoted ASCII string
```

The :SAVE:SETup[:STARt] command saves an oscilloscope setup.

NOTE If a file extension is provided as part of a specified <file_name>, it must be ".scp".

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:FILename" on page 264
- ":RECall:SETup[:STARt]" on page 261

:SAVE:WAVeform[:STARt]

N (see page 530)

Command Syntax

```
:SAVE:WAVeform[:STARt] [<file_name>]

<file_name> ::= quoted ASCII string
```

The :SAVE:WAVeform[:STARt] command saves oscilloscope waveform data to a file.

NOTE If a file extension is provided as part of a specified <file_name>, it must match the extension expected by the format specified in :SAVE:WAVeform:FORMat.

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:WAVeform:FORMat" on page 274
- ":SAVE:WAVeform:LENGth" on page 275
- ":SAVE:FILename" on page 264
- ":RECall:SETup[:STARt]" on page 261

:SAVE:WAVeform:FORMat

N (see page 530)

Command Syntax

```
:SAVE:WAVeform:FORMat <format>

<format> ::= {ALB | ASCiixy | CSV | BINary}
```

The :SAVE:WAVeform:FORMat command sets the waveform data format type:

- ALB — creates an Agilent module binary format file. These files can be viewed offline by the *Agilent Logic Analyzer* application software. The proper file extension for this format is ".alb".
- ASCiixy — creates comma-separated value files for each analog channel that is displayed (turned on). The proper file extension for this format is ".csv".
- CSV — creates one comma-separated value file that contains information for all analog chanels that are displayed (turned on). The proper file extension for this format is ".csv".
- BINary — creates an oscilloscope binary data format file. See the *User's Guide* for a description of this format. The proper file extension for this format is ".bin".

Query Syntax

```
:SAVE:WAVeform:FORMat?
```

The :SAVE:WAVeform:FORMat? query returns the selected waveform data format type.

Return Format

```
<format><NL>

<format> ::= {ALB | ASC | CSV | BIN | NONE}
```

When NONE is returned, it indicates that an image file format is currently selected.

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:WAVeform[:STARt]" on page 273
- ":SAVE:WAVeform:LENGth" on page 275
- ":SAVE:IMAGe:FORMat" on page 268

:SAVE:WAVeform:LENGth

N (see page 530)

Command Syntax

```
:SAVE:WAVeform:LENGth <length>
```

```
<length> ::= 100 to max. length; an integer in NR1 format
```

The :SAVE:WAVeform:LENGth command sets the waveform data length (that is, the number of points saved).

Query Syntax

```
:SAVE:WAVeform:LENGth?
```

The :SAVE:WAVeform:LENGth? query returns the specified waveform data length.

Return Format

```
<length><NL>
```

```
<length> ::= 100 to max. length; an integer in NR1 format
```

See Also

- "Introduction to :SAVE Commands" on page 263
- ":SAVE:WAVeform[:STARt]" on page 273
- ":WAVeform:POINts" on page 413
- ":SAVE:WAVeform:FORMat" on page 274

:SBUS Commands

Control oscilloscope functions associated with the serial decode bus. See "Introduction to :SBUS Commands" on page 277.

Table 53 :SBUS Commands Summary

Command	Query	Options and Query Returns
n/a	:SBUS:CAN:COUNt:ERRor? (see page 278)	<frame_count> ::= integer in NR1 format
n/a	:SBUS:CAN:COUNt:OVERload? (see page 279)	<frame_count> ::= integer in NR1 format
:SBUS:CAN:COUNt:RESet (see page 280)	n/a	n/a
n/a	:SBUS:CAN:COUNt:TOTal? (see page 281)	<frame_count> ::= integer in NR1 format
n/a	:SBUS:CAN:COUNt:UTILization? (see page 282)	<percent> ::= floating-point in NR3 format
:SBUS:DISPlay {{0 | OFF} | {1 | ON}} (see page 283)	:SBUS:DISPlay? (see page 283)	{0 | 1}
:SBUS:IIC:ASIZe <size> (see page 284)	:SBUS:IIC:ASIZe? (see page 284)	<size> ::= {BIT7 | BIT8}
:SBUS:LIN:PARity {{0 | OFF} | {1 | ON}} (see page 285)	:SBUS:LIN:PARity? (see page 285)	{0 | 1}
:SBUS:MODE <mode> (see page 286)	:SBUS:MODE? (see page 286)	<mode> ::= {IIC | SPI | CAN | LIN | FLEXray | UART}
:SBUS:SPI:WIDTh <word_width> (see page 287)	:SBUS:SPI:WIDTh? (see page 287)	<word_width> ::= integer 4-16 in NR1 format
:SBUS:UART:BASE <base> (see page 288)	:SBUS:UART:BASE? (see page 288)	<base> ::= {ASCii | BINary | HEX}
n/a	:SBUS:UART:COUNt:ERRor? (see page 289)	<frame_count> ::= integer in NR1 format
:SBUS:UART:COUNt:RESet (see page 290)	n/a	n/a
n/a	:SBUS:UART:COUNt:RXFRames? (see page 291)	<frame_count> ::= integer in NR1 format

Table 53 :SBUS Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:SBUS:UART:COUNt:TXFRames? (see page 292)	<frame_count> ::= integer in NR1 format
:SBUS:UART:FRAMing <value> (see page 293)	:SBUS:UART:FRAMing? (see page 293)	<value> ::= {OFF | <decimal> | <nondecimal>} <decimal> ::= 8-bit integer from 0-255 (0x00-0xff) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

Introduction to :SBUS Commands

The :SBUS subsystem commands control the serial decode bus viewing, mode, and other options.

NOTE

These commands are only valid on 4 (analog) 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
```

:SBUS:CAN:COUNt:ERRor

N (see page 530)

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 501

See Also

- ":SBUS:CAN:COUNt:RESet" on page 280
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:CAN Commands" on page 324

:SBUS:CAN:COUNt:OVERload

N (see page 530)

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 501

See Also

- ":SBUS:CAN:COUNt:RESet" on page 280
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:CAN Commands" on page 324

:SBUS:CAN:COUNt:RESet

N (see page 530)

Command Syntax

```
:SBUS:CAN:COUNt:RESet
```

Resets the frame counters.

Errors

- "-241, Hardware missing" on page 501

See Also

- ":SBUS:CAN:COUNt:ERRor" on page 278
- ":SBUS:CAN:COUNt:OVERload" on page 279
- ":SBUS:CAN:COUNt:TOTal" on page 281
- ":SBUS:CAN:COUNt:UTILization" on page 282
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:CAN Commands" on page 324

:SBUS:CAN:COUNt:TOTal

N (see page 530)

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 501

See Also

- ":SBUS:CAN:COUNt:RESet" on page 280
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:CAN Commands" on page 324

:SBUS:CAN:COUNt:UTILization

N (see page 530)

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 501

See Also

- ":SBUS:CAN:COUNt:RESet" on page 280
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:CAN Commands" on page 324

:SBUS:DISPlay

N (see page 530)

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 (analog) 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 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":CHANnel<n>:DISPlay" on page 147
- ":VIEW" on page 122
- ":BLANk" on page 94
- ":STATus" on page 119

:SBUS:IIC:ASIZe

N (see page 530)

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 (analog) 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 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":TRIGger:IIC Commands" on page 355

:SBUS:LIN:PARity

N (see page 530)

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.

NOTE

This command is only valid on 4 (analog) channel oscilloscope models when the automotive CAN and LIN serial decode option (Option AMS) has been licensed.

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 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":TRIGger:LIN Commands" on page 364

:SBUS:MODE

N (see page 530)

Command Syntax

```
:SBUS:MODE <mode>
```

```
<mode> ::= {IIC | SPI | CAN | LIN | UART}
```

The :SBUS:MODE command determines the decode mode for the serial bus.

NOTE This command is only valid on 4 (analog) 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 | UART | NONE}
```

Errors

- "-241, Hardware missing" on page 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":TRIGger:MODE" on page 319
- ":TRIGger:IIC Commands" on page 355
- ":TRIGger:SPI Commands" on page 372
- ":TRIGger:CAN Commands" on page 324
- ":TRIGger:LIN Commands" on page 364
- ":TRIGger:UART Commands" on page 387

:SBUS:SPI:WIDTh

N (see page 530)

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 (analog) 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 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:SPI Commands" on page 372

:SBUS:UART:BASE

N (see page 530)

Command Syntax

```
:SBUS:UART:BASE <base>
```

```
<base> ::= {ASCii | BINary | HEX}
```

The :SBUS:UART:BASE command determines the base to use for the UART decode display.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Query Syntax

```
:SBUS:UART:BASE?
```

The :SBUS:UART:BASE? query returns the current UART decode base setting.

Return Format

```
<base><NL>
```

```
<base> ::= {ASCii | BINary | HEX}
```

Errors

- "-241, Hardware missing" on page 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":TRIGger:UART Commands" on page 387

:SBUS:UART:COUNt:ERRor

N (see page 530)

Query Syntax

```
:SBUS:UART:COUNt:ERRor?
```

Returns the UART error frame count.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Return Format

```
<frame_count><NL>

<frame_count> ::= integer in NR1 format
```

Errors

- "-241, Hardware missing" on page 501

See Also

- ":SBUS:UART:COUNt:RESet" on page 290
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:UART Commands" on page 387

:SBUS:UART:COUNt:RESet

N (see page 530)

Command Syntax

```
:SBUS:UART:COUNt:RESet
```

Resets the UART frame counters.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Errors

- "-241, Hardware missing" on page 501

See Also

- ":SBUS:UART:COUNt:ERRor" on page 289
- ":SBUS:UART:COUNt:RXFRames" on page 291
- ":SBUS:UART:COUNt:TXFRames" on page 292
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:UART Commands" on page 387

:SBUS:UART:COUNt:RXFRames

N (see page 530)

Query Syntax

```
:SBUS:UART:COUNt:RXFRames?
```

Returns the UART Rx frame count.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Return Format

```
<frame_count><NL>
```

```
<frame_count> ::= integer in NR1 format
```

Errors

- "-241, Hardware missing" on page 501

See Also

- ":SBUS:UART:COUNt:RESet" on page 290
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:UART Commands" on page 387

:SBUS:UART:COUNt:TXFRames

N (see page 530)

Query Syntax

```
:SBUS:UART:COUNt:TXFRames?
```

Returns the UART Tx frame count.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Return Format

```
<frame_count><NL>
```

```
<frame_count> ::= integer in NR1 format
```

Errors

- "-241, Hardware missing" on page 501

See Also

- ":SBUS:UART:COUNt:RESet" on page 290
- "Introduction to :SBUS Commands" on page 277
- ":SBUS:MODE" on page 286
- ":TRIGger:UART Commands" on page 387

:SBUS:UART:FRAMing

N (see page 530)

Command Syntax

```
:SBUS:UART:FRAMing <value>

<value> ::= {OFF | <decimal> | <nondecimal>}

<decimal> ::= 8-bit integer in decimal from 0-255 (0x00-0xff)

<nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
```

The :SBUS:UART:FRAMing command determines the byte value to use for framing (end of packet) or to turn off framing for UART decode.

NOTE This command is only valid on 4 (analog) channel oscilloscope models when the UART/RS-232 triggering and serial decode option (Option 232) has been licensed.

Query Syntax

```
:SBUS:UART:FRAMing?
```

The :SBUS:UART:FRAMing? query returns the current UART decode base setting.

Return Format

```
<value><NL>

<value> ::= {OFF | <decimal>}

<decimal> ::= 8-bit integer in decimal from 0-255
```

Errors

- "-241, Hardware missing" on page 501

See Also

- "Introduction to :SBUS Commands" on page 277
- ":TRIGger:UART Commands" on page 387

:SYSTem Commands

Control basic system functions of the oscilloscope. See "Introduction to :SYSTem Commands" on page 294.

Table 54 :SYSTem Commands Summary

Command	Query	Options and Query Returns

:SYSTem:DATE <date> (see page 295)	:SYSTem:DATE? (see page 295)	<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 296)	n/a	<string> ::= up to 254 characters as a quoted ASCII string
n/a	:SYSTem:ERRor? (see page 297)	<error> ::= an integer error code <error string> ::= quoted ASCII string. See Error Messages (see page 499).
:SYSTem:LOCK (see page 298)	:SYSTem:LOCK? (see page 298)	<value> ::= {ON | OFF}
:SYSTem:SETup <setup_data> (see page 299)	:SYSTem:SETup? (see page 299)	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:TIME <time> (see page 301)	:SYSTem:TIME? (see page 301)	<time> ::= hours,minutes,seconds in NR1 format

Introduction to :SYSTem Commands

SYSTem subsystem commands enable writing messages to the display, setting and reading both the time and the date, querying for errors, and saving and recalling setups.

:SYSTem:DATE

N (see page 530)

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 294
- ":SYSTem:TIME" on page 301

:SYSTem:DSP

N (see page 530)

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 294

:SYSTem:ERRor

C (see page 530)

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 499.

See Also

- "Introduction to :SYSTem Commands" on page 294
- "*ESR (Standard Event Status Register)" on page 68
- "*CLS (Clear Status)" on page 65

:SYSTem:LOCK

N (see page 530)

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 294

:SYSTem:SETup

C (see page 530)

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 294
- "*LRN (Learn Device Setup)" on page 71

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_UI1)
CheckForInstrumentErrors   ' After reading query results.

' Output setup string to a file:
Dim strPath As String
strPath = "c:\scope\config\setup.dat"

' Open file for output.
Close #1   ' If #1 is open, close it.
Open strPath For Binary Access Write Lock Write As #1
Put #1, , varQueryResult   ' Write data.
Close #1   ' Close file.

' RESTORE_SYSTEM_SETUP - Read the setup string from a file and
' write it back to the oscilloscope.
Dim varSetupString As Variant
strPath = "c:\scope\config\setup.dat"

' Open file for input.
Open strPath For Binary Access Read As #1
Get #1, , varSetupString   ' Read data.
Close #1   ' Close file.
```

```
' Write setup string back to oscilloscope using ":SYSTEM:SETUP"
' command:
myScope.WriteIEEEBlock ":SYSTEM:SETUP ", varSetupString
CheckForInstrumentErrors
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:SYSTem:TIME

N (see page 530)

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 294
- ":SYSTem:DATE" on page 295

:TIMebase Commands

Control all horizontal sweep functions. See "Introduction to :TIMebase Commands" on page 302.

Table 55 :TIMebase Commands Summary

Command	Query	Options and Query Returns
:TIMebase:MODE <value> (see page 304)	:TIMebase:MODE? (see page 304)	<value> ::= {MAIN | WINDow | XY | ROLL}
:TIMebase:POSition <pos> (see page 305)	:TIMebase:POSition? (see page 305)	<pos> ::= time from the trigger event to the display reference point in NR3 format
:TIMebase:RANGe <range_value> (see page 306)	:TIMebase:RANGe? (see page 306)	<range_value> ::= 10 ns through 500 s in NR3 format
:TIMebase:REFerence {LEFT | CENTer | RIGHt} (see page 307)	:TIMebase:REFerence? (see page 307)	<return_value> ::= {LEFT | CENTer | RIGHt}
:TIMebase:SCALe <scale_value> (see page 308)	:TIMebase:SCALe? (see page 308)	<scale_value> ::= scale value in seconds in NR3 format
:TIMebase:VERNier {{0 | OFF} | {1 | ON}} (see page 309)	:TIMebase:VERNier? (see page 309)	{0 | 1}
:TIMebase:WINDow:POSition <pos> (see page 310)	:TIMebase:WINDow:POSition? (see page 310)	<pos> ::= time from the trigger event to the delayed view reference point in NR3 format
:TIMebase:WINDow:RANGe <range_value> (see page 311)	:TIMebase:WINDow:RANGe? (see page 311)	<range value> ::= range value in seconds in NR3 format for the delayed window
:TIMebase:WINDow:SCALe <scale_value> (see page 312)	:TIMebase:WINDow:SCALe? (see page 312)	<scale_value> ::= scale value in seconds in NR3 format for the delayed window

Introduction to :TIMebase Commands

The TIMebase subsystem commands control the horizontal (X-axis) functions and set the oscilloscope to X-Y mode (where channel 1 becomes the X input and channel 2 becomes the Y input). The time per division, delay, vernier control, and reference can be controlled for the main and window (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

C (see page 530)

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 302
- "*RST (Reset)" on page 75
- ":TIMebase:RANGe" on page 306
- ":TIMebase:POSition" on page 305
- ":TIMebase:REFerence" on page 307

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 578

:TIMebase:POSition

C (see page 530)

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 302
- ":TIMebase:REFerence" on page 307
- ":TIMebase:RANGe" on page 306
- ":TIMebase:SCALe" on page 308
- ":TIMebase:WINDow:POSition" on page 310
- ":TIMebase:DELay" on page 494

:TIMebase:RANGe

C (see page 530)

Command Syntax

```
:TIMebase:RANGe <range_value>
```

```
<range_value> ::= 10 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> ::= 10 ns through 500 s in NR3 format
```

See Also

- "Introduction to :TIMebase Commands" on page 302
- ":TIMebase:MODE" on page 304
- ":TIMebase:SCALe" on page 308
- ":TIMebase:WINDow:RANGe" on page 311

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 578

:TIMebase:REFerence

C (see page 530)

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 302
- ":TIMebase:MODE" on page 304

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 578

:TIMebase:SCALe

N (see page 530)

Command Syntax

```
:TIMebase:SCALe <scale_value>
```

```
<scale_value> ::= 1 ns through 50 s in NR3 format
```

The :TIMebase:SCALe command sets the horizontal scale or units per division for the main window.

Query Syntax

```
:TIMebase:SCALe?
```

The :TIMebase:SCALe? query returns the current horizontal scale setting in seconds per division for the main window.

Return Format

```
<scale_value><NL>
```

```
<scale_value> ::= 1 ns through 50 s in NR3 format
```

See Also

- "Introduction to :TIMebase Commands" on page 302
- ":TIMebase:RANGe" on page 306
- ":TIMebase:WINDow:SCALe" on page 312
- ":TIMebase:WINDow:RANGe" on page 311

:TIMebase:VERNier

N (see page 530)

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 302

:TIMebase:WINDow:POSition

C (see page 530)

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 302
- ":TIMebase:MODE" on page 304
- ":TIMebase:POSition" on page 305
- ":TIMebase:RANGe" on page 306
- ":TIMebase:SCALe" on page 308
- ":TIMebase:WINDow:RANGe" on page 311
- ":TIMebase:WINDow:SCALe" on page 312

:TIMebase:WINDow:RANGe

C (see page 530)

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 302
- ":TIMebase:RANGe" on page 306
- ":TIMebase:POSition" on page 305
- ":TIMebase:SCALe" on page 308

:TIMebase:WINDow:SCALe

N (see page 530)

Command Syntax

```
: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

```
: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 302
- ":TIMebase:RANGe" on page 306
- ":TIMebase:POSition" on page 305
- ":TIMebase:SCALe" on page 308
- ":TIMebase:WINDow:RANGe" on page 311

:TRIGger Commands

Control the trigger modes and parameters for each trigger type. See:

- "Introduction to :TRIGger Commands" on page 313
- "General :TRIGger Commands" on page 316
- ":TRIGger:CAN Commands" on page 324
- ":TRIGger:DURation Commands" on page 335
- ":TRIGger[:EDGE] Commands" on page 341
- ":TRIGger:GLITch Commands" on page 347 (Pulse Width trigger)
- ":TRIGger:IIC Commands" on page 355
- ":TRIGger:LIN Commands" on page 364
- ":TRIGger:SPI Commands" on page 372
- ":TRIGger:TV Commands" on page 381
- ":TRIGger:UART Commands" on page 387

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 323) 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 319).

- **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.

NOTE

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.
- **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.
- **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 º division of sync amplitude with any analog channel as the trigger source.
- **UART/RS-232 triggering** (with Option 232) lets you trigger on RS-232 serial data.

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 56 General :TRIGger Commands Summary

Command	Query	Options and Query Returns
:TRIGger:HFReject {{0 | OFF} | {1 | ON}} (see page 317)	:TRIGger:HFReject? (see page 317)	{0 | 1}
:TRIGger:HOLDoff <holdoff_time> (see page 318)	:TRIGger:HOLDoff? (see page 318)	<holdoff_time> ::= 60 ns to 10 s in NR3 format

:TRIGger:MODE <mode> (see page 319)	:TRIGger:MODE? (see page 319)	<mode> ::= {EDGE | GLITch | PATTern | DURation | TV}
		<return_value> ::= {<mode> | <none>}
		<none> ::= query returns "NONE" if the :TIMebase:MODE is ROLL or XY

:TRIGger:NREJect {{0 | OFF} | {1 | ON}} (see page 320)	:TRIGger:NREJect? (see page 320)	{0 | 1}
:TRIGger:PATTern <value>, <mask> [,<edge source>,<edge>] (see page 321)	:TRIGger:PATTern? (see page 321)	<value> ::= integer in NR1 format or <string> <mask> ::= integer in NR1 format or <string> <string> ::= "0xnn"; n ::= {0,..,9 | A,..,F} (# bits = # channels) <edge source> ::= {CHANnel<n> | EXTernal | NONE} <edge> ::= {POSitive | NEGative} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SWEep <sweep> (see page 323)	:TRIGger:SWEep? (see page 323)	<sweep> ::= {AUTO | NORMal}

:TRIGger:HFReject

C (see page 530)

Command Syntax

```
:TRIGger:HFReject <value>
```

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

The :TRIGger:HFReject command turns the high frequency reject filter off and on. The high frequency reject filter adds a 50 kHz low-pass filter in the trigger path to remove high frequency components from the trigger waveform. Use this filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.

Query Syntax

```
:TRIGger:HFReject?
```

The :TRIGger:HFReject? query returns the current high frequency reject filter mode.

Return Format

```
<value><NL>
```

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

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger[:EDGE]:REJect" on page 344

:TRIGger:HOLDoff

C (see page 530)

Command Syntax

```
: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

```
:TRIGger:HOLDoff?
```

The :TRIGger:HOLDoff? query returns the holdoff time value for the current trigger mode.

Return Format

```
<holdoff_time><NL>
```

```
<holdoff_time> ::= the holdoff time value in seconds in NR3 format.
```

See Also

- "Introduction to :TRIGger Commands" on page 313

:TRIGger:MODE

C (see page 530)

Command Syntax

```
:TRIGger:MODE <mode>

1234567890123456789012345678901234567890123456789012345678901234567890
<mode> ::= {EDGE | GLITch | PATTern | CAN | DURation | IIC | LIN | SPI
           | TV | USB | FLEXray | UART}
```

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 | GLITch | PATTern | CAN | DURation | IIC
           | LIN | SPI | TV | USB | FLEXray | UART}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:SWEep" on page 323
- ":TIMebase:MODE" on page 304

Example Code

```
' TRIGGER_MODE - Set the trigger mode to EDGE, GLITch, PATTern, CAN,
' DURation, IIC, LIN, SPI, TV, USB, FLEXray, or UART.

' Set the trigger mode to EDGE.
myScope.WriteString ":TRIGGER:MODE EDGE"
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:TRIGger:NREJect

C (see page 530)

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 313

:TRIGger:PATTern

C (see page 530)

Command Syntax

```
:TRIGger:PATTern <pattern>

<pattern> ::= <value>, <mask> [, <edge source>, <edge>]

<value> ::= integer in NR1 format or <string>

<mask> ::= integer in NR1 format or <string>

<string> ::= "0xnn"; n ::= {0,..,9 | A,..,F}
             (# bits = # channels, see following table)

<edge source> ::= {CHANnel<n> | EXTernal | NONE}

<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:

Oscilloscope Models	Value and Mask Bit Assignments
4 analog channels	Bits 0 through 3 - analog channels 1 through 4. Bit 4 - external trigger.
2 analog channels	Bits 0 and 1 - analog channels 1 and 2. Bit 4 - external trigger.

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:PATTern?
```

The :TRIGger:PATTern? 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 313

- ":TRIGger:MODE" on page 319

:TRIGger:SWEep

C (see page 530)

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 313

:TRIGger:CAN Commands

Table 57 :TRIGger:CAN Commands Summary

Command	Query	Options and Query Returns
:TRIGger:CAN:PATTern:DATA <value>, <mask> (see page 326)	:TRIGger:CAN:PATTern:DATA? (see page 326)	<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 327)	:TRIGger:CAN:PATTern:DATA:LENGth? (see page 327)	<length> ::= integer from 1 to 8 in NR1 format (with Option AMS)
:TRIGger:CAN:PATTern:ID <value>, <mask> (see page 328)	:TRIGger:CAN:PATTern:ID? (see page 328)	<value> ::= 32-bit integer in decimal, <nondecimal>, or <string> (with Option AMS) <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
:TRIGger:CAN:PATTern:ID:MODE <value> (see page 329)	:TRIGger:CAN:PATTern:ID:MODE? (see page 329)	<value> ::= {STANdard | EXTended} (with Option AMS)
:TRIGger:CAN:SAMPlepoint <value> (see page 330)	:TRIGger:CAN:SAMPlepoint? (see page 330)	<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
:TRIGger:CAN:SIGNal:BAUDrate <baudrate> (see page 331)	:TRIGger:CAN:SIGNal:BAUDrate? (see page 331)	<baudrate> ::= {10000 | 20000 | 33300 | 50000 | 62500 | 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000}

Table 57 :TRIGger:CAN Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:CAN:SOURce <source> (see page 332)	:TRIGger:CAN:SOURce? (see page 332)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 |} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:CAN:TRIGger <condition> (see page 333)	:TRIGger:CAN:TRIGger? (see page 334)	<condition> ::= {SOF} (without Option AMS) <condition> ::= {SOF | DATA | ERRor | IDData | IDEither | IDRemote | ALLerrors | OVERload | ACKerror} (with Option AMS)

:TRIGger:CAN:PATTern:DATA

N (see page 530)

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?
```

The :TRIGger:CAN:PATTern:DATA? query returns the current settings of the specified CAN data pattern resource.

Return Format

```
<value>, <mask><NL> in nondecimal format
```

Errors

- "-241, Hardware missing" on page 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
- ":TRIGger:CAN:PATTern:ID" on page 328

:TRIGger:CAN:PATTern:DATA:LENGth

N (see page 530)

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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:CAN:PATTern:DATA" on page 326
- ":TRIGger:CAN:SOURce" on page 332

:TRIGger:CAN:PATTern:ID

N (see page 530)

Command Syntax

```
:TRIGger:CAN:PATTern: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:PATTern: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:PATTern: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:PATTern: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:PATTern:ID?
```

The :TRIGger:CAN:PATTern: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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:CAN:PATTern:ID:MODE" on page 329
- ":TRIGger:CAN:PATTern:DATA" on page 326

:TRIGger:CAN:PATTern:ID:MODE

N (see page 530)

Command Syntax

```
:TRIGger:CAN:PATTern:ID:MODE <value>

<value> ::= {STANdard | EXTended}
```

The :TRIGger:CAN:PATTern:ID:MODE command sets the CAN identifier mode. STANdard selects the standard 11-bit identifier. EXTended selects the extended 29-bit identifier. The CAN identifier is set by the :TRIGger:CAN:PATTern:ID 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: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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:PATTern:DATA" on page 326
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
- ":TRIGger:CAN:PATTern:ID" on page 328

:TRIGger:CAN:SAMPlepoint

N (see page 530)

Command Syntax

```
: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

```
:TRIGger:CAN:SAMPlepoint?
```

The :TRIGger:CAN:SAMPlepoint? query returns the current CAN sample point setting.

Return Format

```
<value><NL>
```

```
<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:TRIGger" on page 333

:TRIGger:CAN:SIGNal:BAUDrate

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:TRIGger" on page 333
- ":TRIGger:CAN:SIGNal:DEFinition" on page 496
- ":TRIGger:CAN:SOURce" on page 332

:TRIGger:CAN:SOURce

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:TRIGger" on page 333
- ":TRIGger:CAN:SIGNal:DEFinition" on page 496

:TRIGger:CAN:TRIGger

N (see page 530)

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:

Remote <condition> parameter	**Front-panel Trigger on: softkey selection (softkey text - softkey popup text)**
SOF	SOF - Start of Frame
DATA	Id & Data - Data Frame Id and Data
ERRor	Error - Error frame
IDData	Id & ~RTR - Data Frame Id (~RTR)
IDEither	Id - Remote or Data Frame Id
IDRemote	Id & RTR - Remote Frame Id (RTR)
ALLerrors	All Errors - All Errors
OVERload	Overload - Overload Frame
ACKerror	Ack Error - Acknowledge Error

CAN Id specification is set by the :TRIGger:CAN:PATTern:ID and:TRIGger:CAN:PATTern:ID:MODE commands.

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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:PATTern:DATA" on page 326
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
- ":TRIGger:CAN:PATTern:ID" on page 328
- ":TRIGger:CAN:PATTern:ID:MODE" on page 329
- ":TRIGger:CAN:SIGNal:DEFinition" on page 496
- ":TRIGger:CAN:SOURce" on page 332

:TRIGger:DURation Commands

Table 58 :TRIGger:DURation Commands Summary

Command	Query	Options and Query Returns
:TRIGger:DURation:GREaterthan <greater than time>[suffix] (see page 336)	:TRIGger:DURation:GREaterthan? (see page 336)	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:DURation:LESSthan <less than time>[suffix] (see page 337)	:TRIGger:DURation:LESSthan? (see page 337)	<less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:DURation:PATTern <value>, <mask> (see page 338)	:TRIGger:DURation:PATTern? (see page 338)	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= ""0xnnnnnn"" n ::= {0,..,9 | A,..,F}
:TRIGger:DURation:QUALifier <qualifier> (see page 339)	:TRIGger:DURation:QUALifier? (see page 339)	<qualifier> ::= {GREaterthan | LESSthan | INRange | OUTRange | TIMeout}
:TRIGger:DURation:RANGe <greater than time>[suffix], <less than time>[suffix] (see page 340)	:TRIGger:DURation:RANGe? (see page 340)	<greater than time> ::= min duration from 10 ns to 9.99 seconds in NR3 format <less than time> ::= max duration from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}

:TRIGger:DURation:GREaterthan

N (see page 530)

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 313
- ":TRIGger:DURation:PATTern" on page 338
- ":TRIGger:DURation:QUALifier" on page 339
- ":TRIGger:MODE" on page 319

:TRIGger:DURation:LESSthan

N (see page 530)

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 313
- ":TRIGger:DURation:PATTern" on page 338
- ":TRIGger:DURation:QUALifier" on page 339
- ":TRIGger:MODE" on page 319

:TRIGger:DURation:PATTern

N (see page 530)

Command Syntax

```
:TRIGger:DURation:PATTern <value>, <mask>

<value> ::= integer or <string>

<mask> ::= integer or <string>

<string> ::= "0xnnnnnnn"; 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:

Oscilloscope Models	Value and Mask Bit Assignments
4 analog channels	Bits 0 through 3 - analog channels 1 through 4. Bit 4 - external trigger.
2 analog channels	Bits 0 and 1 - analog channels 1 and 2. Bit 4 - external trigger.

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 313
- ":TRIGger:PATTern" on page 321

:TRIGger:DURation:QUALifier

N (see page 530)

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 313
- ":TRIGger:DURation:GREaterthan" on page 336
- ":TRIGger:DURation:LESSthan" on page 337
- ":TRIGger:DURation:RANGe" on page 340

:TRIGger:DURation:RANGe

N (see page 530)

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 313
- ":TRIGger:DURation:PATTern" on page 338
- ":TRIGger:DURation:QUALifier" on page 339
- ":TRIGger:MODE" on page 319

:TRIGger[:EDGE] Commands

Table 59 :TRIGger[:EDGE] Commands Summary

Command	Query	Options and Query Returns
:TRIGger[:EDGE]:COUPling {AC | DC | LF} (see page 342)	:TRIGger[:EDGE]:COUPling? (see page 342)	{AC | DC | LF}
:TRIGger[:EDGE]:LEVel <level> [,<source>] (see page 343)	:TRIGger[:EDGE]:LEVel? [<source>] (see page 343)	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. <source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger[:EDGE]:REJect {OFF | LF | HF} (see page 344)	:TRIGger[:EDGE]:REJect? (see page 344)	{OFF | LF | HF}
:TRIGger[:EDGE]:SLOPe <polarity> (see page 345)	:TRIGger[:EDGE]:SLOPe? (see page 345)	<polarity> ::= {POSitive | NEGative | EITHer | ALTernate}
:TRIGger[:EDGE]:SOURce <source> (see page 346)	:TRIGger[:EDGE]:SOURce? (see page 346)	<source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format

:TRIGger[:EDGE]:COUPling

C (see page 530)

Command Syntax

```
:TRIGger[:EDGE]:COUPling <coupling>

<coupling> ::= {AC | DC | LFReject}
```

The :TRIGger[:EDGE]:COUPling command sets the input coupling for the selected trigger sources. The coupling can be set to AC, DC, or LFReject.

- AC coupling places a high-pass filter (10 Hz for analog channels, and 3.5 Hz for all External trigger inputs) in the trigger path, removing dc offset voltage from the trigger waveform. Use AC coupling to get a stable edge trigger when your waveform has a large dc offset.
- LFReject coupling places a 50 KHz high-pass filter in the trigger path.
- DC coupling allows dc and ac signals into the trigger path.

NOTE The :TRIGger[:EDGE]:COUPling and the :TRIGger[:EDGE]:REJect selections are coupled. Changing the setting of the :TRIGger[:EDGE]:REJect can change the COUPling setting.

Query Syntax

```
:TRIGger[:EDGE]:COUPling?
```

The :TRIGger[:EDGE]:COUPling? query returns the current coupling selection.

Return Format

```
<coupling><NL>

<coupling> ::= {AC | DC | LFR}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger[:EDGE]:REJect" on page 344

:TRIGger[:EDGE]:LEVel

C (see page 530)

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

<source> ::= {CHANnel<n> | EXTernal}

<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 313
- ":TRIGger[:EDGE]:SOURce" on page 346

:TRIGger[:EDGE]:REJect

C (see page 530)

Command Syntax

```
:TRIGger[:EDGE]:REJect <reject>
```

```
<reject> ::= {OFF | LFReject | HFReject}
```

The :TRIGger[:EDGE]:REJect command turns the low-frequency or high-frequency reject filter on or off. You can turn on one of these filters at a time.

- The high frequency reject filter adds a 50 kHz low-pass filter in the trigger path to remove high frequency components from the trigger waveform. Use the high frequency reject filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- The low frequency reject filter adds a 50 kHz high-pass filter in series with the trigger waveform to remove any unwanted low frequency components from a trigger waveform, such as power line frequencies, that can interfere with proper triggering.

NOTE The :TRIGger[:EDGE]:REJect and the :TRIGger[:EDGE]:COUPling selections are coupled. Changing the setting of the :TRIGger[:EDGE]:COUPling can change the COUPling setting.

Query Syntax

```
:TRIGger[:EDGE]:REJect?
```

The :TRIGger[:EDGE]:REJect? query returns the current status of the reject filter.

Return Format

```
<reject><NL>
```

```
<reject> ::= {OFF | LFR | HFR}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:HFReject" on page 317
- ":TRIGger[:EDGE]:COUPling" on page 342

:TRIGger[:EDGE]:SLOPe

C (see page 530)

Command Syntax

```
:TRIGger[:EDGE]:SLOPe <slope>
```

```
<slope> ::= {NEGative | POSitive | 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 | ALT}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:TV:POLarity" on page 384

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 578

:TRIGger[:EDGE]:SOURce

C (see page 530)

Command Syntax

```
:TRIGger[:EDGE]:SOURce <source>

<source> ::= {CHANnel<n> | EXTernal | LINE}

<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}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319

Example Code

```
   ' TRIGGER_EDGE_SOURCE - Selects the channel that actually produces th
e
   ' 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 578

:TRIGger:GLITch Commands

Table 60 :TRIGger:GLITch Commands Summary

Command	Query	Options and Query Returns
:TRIGger:GLITch:GREaterthan <greater than time>[suffix] (see page 348)	:TRIGger:GLITch:GREaterthan? (see page 348)	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:LESSthan <less than time>[suffix] (see page 349)	:TRIGger:GLITch:LESSthan? (see page 349)	<less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:LEVel <level> [<source>] (see page 350)	:TRIGger:GLITch:LEVel? (see page 350)	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. <source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:GLITch:POLarity <polarity> (see page 351)	:TRIGger:GLITch:POLarity? (see page 351)	<polarity> ::= {POSitive | NEGative}
:TRIGger:GLITch:QUALifier <qualifier> (see page 352)	:TRIGger:GLITch:QUALifier? (see page 352)	<qualifier> ::= {GREaterthan | LESSthan | RANGe}
:TRIGger:GLITch:RANGe <greater than time>[suffix], <less than time>[suffix] (see page 353)	:TRIGger:GLITch:RANGe? (see page 353)	<greater than time> ::= start time from 10 ns to 9.99 seconds in NR3 format <less than time> ::= stop time from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:SOURce <source> (see page 354)	:TRIGger:GLITch:SOURce? (see page 354)	<source> ::= {CHANnel<n> | EXTernal} <n> ::= 1-2 or 1-4 in NR1 format

:TRIGger:GLITch:GREaterthan

N (see page 530)

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 313
- ":TRIGger:GLITch:SOURce" on page 354
- ":TRIGger:GLITch:QUALifier" on page 352
- ":TRIGger:MODE" on page 319

:TRIGger:GLITch:LESSthan

N (see page 530)

Command Syntax

```
: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

```
:TRIGger:GLITch:LESSthan?
```

The :TRIGger:GLITch:LESSthan? query returns the pulse width duration time for :TRIGger:GLITch:SOURce.

Return Format

```
<less_than_time><NL>

<less_than_time> ::= a 32-bit floating-point number in NR3 format.
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:GLITch:SOURce" on page 354
- ":TRIGger:GLITch:QUALifier" on page 352
- ":TRIGger:MODE" on page 319

:TRIGger:GLITch:LEVel

N (see page 530)

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

<source> ::= {CHANnel<n> | EXTernal}

<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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:GLITch:SOURce" on page 354

:TRIGger:GLITch:POLarity

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:GLITch:SOURce" on page 354

:TRIGger:GLITch:QUALifier

N (see page 530)

Command Syntax

```
:TRIGger:GLITch:QUALifier <operator>

<operator> ::= {GREaterthan | LESSthan | RANGe}
```

This command sets the mode of operation of the glitch pulse width trigger. The oscilloscope can trigger on a pulse width that is greater than a time value, less than a time value, or within a range of time values.

Query Syntax

```
:TRIGger:GLITch:QUALifier?
```

The :TRIGger:GLITch:QUALifier? query returns the glitch pulse width qualifier.

Return Format

```
<operator><NL>

<operator> ::= {GRE | LESS | RANG}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:GLITch:SOURce" on page 354
- ":TRIGger:MODE" on page 319

:TRIGger:GLITch:RANGe

N (see page 530)

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 313
- ":TRIGger:GLITch:SOURce" on page 354
- ":TRIGger:GLITch:QUALifier" on page 352
- ":TRIGger:MODE" on page 319

:TRIGger:GLITch:SOURce

N (see page 530)

Command Syntax

```
:TRIGger:GLITch:SOURce <source>
```

```
<source> ::= {CHANnel<n> | EXTernal}
```

```
<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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:GLITch:LEVel" on page 350
- ":TRIGger:GLITch:POLarity" on page 351
- ":TRIGger:GLITch:QUALifier" on page 352
- ":TRIGger:GLITch:RANGe" on page 353

Example Code

- "Example Code" on page 346

:TRIGger:IIC Commands

Table 61 :TRIGger:IIC Commands Summary

Command	Query	Options and Query Returns
:TRIGger:IIC:PATTern:ADDRess <value> (see page 356)	:TRIGger:IIC:PATTern:ADDRess? (see page 356)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}
:TRIGger:IIC:PATTern:DATA <value> (see page 357)	:TRIGger:IIC:PATTern:DATA? (see page 357)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}
:TRIGger:IIC:PATTern:DATa2 <value> (see page 358)	:TRIGger:IIC:PATTern:DATa2? (see page 358)	<value> ::= integer or <string> <string> ::= "0xnn" n ::= {0,..,9 | A,..,F}
:TRIGger:IIC[:SOURce]:CLOCk <source> (see page 359)	:TRIGger:IIC[:SOURce]:CLOCk? (see page 359)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 } for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC[:SOURce]:DATA <source> (see page 360)	:TRIGger:IIC[:SOURce]:DATA? (see page 360)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15 } for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC:TRIGger:QUALifier <value> (see page 361)	:TRIGger:IIC:TRIGger:QUALifier? (see page 361)	<value> ::= {EQUal | NOTequal | LESSthan | GREaterthan}
:TRIGger:IIC:TRIGger[:TYPE] <type> (see page 362)	:TRIGger:IIC:TRIGger[:TYPE]? (see page 362)	<type> ::= {STARt | STOP | READ7 | READEprom | WRITe7 | WRITe10 | NACKnowledge | ANACknowledge | R7Data2 | W7Data2 | RESTart}

:TRIGger:IIC:PATTern:ADDRess

N (see page 530)

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 313
- ":TRIGger:IIC:PATTern:DATA" on page 357
- ":TRIGger:IIC:PATTern:DATa2" on page 358
- ":TRIGger:IIC:TRIGger[:TYPE]" on page 362

:TRIGger:IIC:PATTern:DATA

N (see page 530)

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 313
- ":TRIGger:IIC:PATTern:ADDRess" on page 356
- ":TRIGger:IIC:PATTern:DATa2" on page 358
- ":TRIGger:IIC:TRIGger[:TYPE]" on page 362

:TRIGger:IIC:PATTern:DATa2

N (see page 530)

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?
```

The :TRIGger:IIC:PATTern:DATa2? query returns the current pattern for IIC data 2.

Return Format

```
<value><NL>
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:IIC:PATTern:ADDRess" on page 356
- ":TRIGger:IIC:PATTern:DATA" on page 357
- ":TRIGger:IIC:TRIGger[:TYPE]" on page 362

:TRIGger:IIC:SOURce:CLOCk

N (see page 530)

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 313
- ":TRIGger:IIC:SOURce:DATA" on page 360

:TRIGger:IIC:SOURce:DATA

N (see page 530)

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 313
- ":TRIGger:IIC:SOURce:CLOCk" on page 359

:TRIGger:IIC:TRIGger:QUALifier

N (see page 530)

Command Syntax

```
:TRIGger:IIC:TRIGger:QUALifier <value>
```

```
<value> ::= {EQUal | NOTequal | LESSthan | GREaterthan}
```

The :TRIGger:IIC:TRIGger:QUALifier command sets the IIC data qualifier when TRIGger:IIC:TRIGger[:TYPE] is set to READEprom.

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 | NOTequal | LESSthan | GREaterthan}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:IIC:TRIGger[:TYPE]" on page 362

:TRIGger:IIC:TRIGger[:TYPE]

N (see page 530)

Command Syntax

```
:TRIGger:IIC:TRIGger[:TYPE] <value>

<value> ::= {STARt | STOP | READ7 | READEprom | 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).
- READEprom — EEPROM data read.
- WRITe7 — 7-bit address frame containing (Start:Address7:Write:Ack:Data). The value WRITe is also accepted for WRITe7.
- W7Data2 — 7-bit address frame containing (Start:Address7:Write:Ack:Data:Ack:Data2).
- WRITe10 — 10-bit address frame containing (Start:Address byte1:Write:Ack:Address byte 2:Data).
- NACKnowledge — Missing acknowledge.
- ANACknowledge — Address with no acknowledge.
- RESTart — Another start condition occurs before a stop condition.

NOTE The short form of READ7 (READ7), READEprom (READE), WRITe7 (WRIT7), and WRITe10 (WRIT10) do not follow the defined Long Form to Short Form Truncation Rules (see page 532).

Query Syntax

```
:TRIGger:IIC:TRIGger[:TYPE]?
```

The :TRIGger:IIC:TRIGger[:TYPE]? query returns the current IIC trigger type value.

Return Format

```
<value><NL>

<value> ::= {STAR | STOP | READ7 | READE | WRIT7 | WRIT10 | NACK | ANAC
            | R7D2 | W7D2 | REST}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319

- ":TRIGger:IIC:PATTern:ADDRess" on page 356
- ":TRIGger:IIC:PATTern:DATA" on page 357
- ":TRIGger:IIC:PATTern:DATa2" on page 358
- ":TRIGger:IIC:TRIGger:QUALifier" on page 361
- "Long Form to Short Form Truncation Rules" on page 532

:TRIGger:LIN Commands

Table 62 :TRIGger:LIN Commands Summary

Command	Query	Options and Query Returns
:TRIGger:LIN:ID <value> (see page 365)	:TRIGger:LIN:ID? (see page 365)	<value> ::= 7-bit integer in decimal, <nondecimal>, or <string> from 0-63 or 0x00-0x3f (with Option AMS) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary <string> ::= "0xnn" where n ::= {0,..,9 | A,..,F} for hexadecimal
:TRIGger:LIN:SAMPlepoint <value> (see page 366)	:TRIGger:LIN:SAMPlepoint? (see page 366)	<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
:TRIGger:LIN:SIGNal:BAUDrate <baudrate> (see page 367)	:TRIGger:LIN:SIGNal:BAUDrate? (see page 367)	<baudrate> ::= {2400 | 9600 | 19200}
:TRIGger:LIN:SOURce <source> (see page 368)	:TRIGger:LIN:SOURce? (see page 368)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:LIN:STANdard <std> (see page 369)	:TRIGger:LIN:STANdard? (see page 369)	<std> ::= {LIN13 | LIN20}
:TRIGger:LIN:SYNCbreak <value> (see page 370)	:TRIGger:LIN:SYNCbreak? (see page 370)	<value> ::= integer = {11 | 12 | 13}
:TRIGger:LIN:TRIGger <condition> (see page 371)	:TRIGger:LIN:TRIGger? (see page 371)	<condition> ::= {SYNCbreak} (without Option AMS) <condition> ::= {SYNCbreak | ID} (with Option AMS)

:TRIGger:LIN:ID

N (see page 530)

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 | A,..,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary

<string> ::= "0xnn" where n ::= {0,..,9 | 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.

Setting the ID to a value of "-1" results in "0xXX" which is equivalent to all IDs.

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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:TRIGger" on page 371
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:LIN:SAMPlepoint

N (see page 530)

Command Syntax

```
:TRIGger:LIN:SAMPlepoint <value>
```

```
<value><NL>
```

```
<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} 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

```
:TRIGger:LIN:SAMPlepoint?
```

The :TRIGger:LIN:SAMPlepoint? query returns the current LIN sample point setting.

Return Format

```
<value><NL>
```

```
<value> ::= {60 | 62.5 | 68 | 70 | 75 | 80 | 87.5} in NR3 format
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:TRIGger" on page 371

:TRIGger:LIN:SIGNal:BAUDrate

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:TRIGger" on page 371
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:LIN:SOURce

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:TRIGger" on page 371
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497

:TRIGger:LIN:STANdard

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:LIN:SYNCbreak

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:LIN:TRIGger

N (see page 530)

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 501

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:SPI Commands

Table 63 :TRIGger:SPI Commands Summary

Command	Query	Options and Query Returns
:TRIGger:SPI:CLOCk:SLOPe <slope> (see page 373)	:TRIGger:SPI:CLOCk:SLOPe? (see page 373)	<slope> ::= {NEGative | POSitive}
:TRIGger:SPI:CLOCk:TIMeout <time_value> (see page 374)	:TRIGger:SPI:CLOCk:TIMeout? (see page 374)	<time_value> ::= time in seconds in NR1 format
:TRIGger:SPI:FRAMing <value> (see page 375)	:TRIGger:SPI:FRAMing? (see page 375)	<value> ::= {CHIPselect | NOTChipselect | TIMeout}
:TRIGger:SPI:PATTern:DATA <value>, <mask> (see page 376)	:TRIGger:SPI:PATTern:DATA? (see page 376)	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= "0xnnnnnn" where n ::= {0,..,9 | A,..,F}
:TRIGger:SPI:PATTern:WIDTh <width> (see page 377)	:TRIGger:SPI:PATTern:WIDTh? (see page 377)	<width> ::= integer from 4 to 32 in NR1 format
:TRIGger:SPI:SOURce:CLOCk <source> (see page 378)	:TRIGger:SPI:SOURce:CLOCk? (see page 378)	<value> ::= {CHANnel<n> | EXTernal} for the DSO models <value> ::= {CHANnel<n> | DIGital0,..,DIGital15} for the MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SPI:SOURce:DATA <source> (see page 379)	:TRIGger:SPI:SOURce:DATA? (see page 379)	<value> ::= {CHANnel<n> | EXTernal} for the DSO models <value> ::= {CHANnel<n> | DIGital0,..,DIGital15} for the MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SPI:SOURce:FRAMe <source> (see page 380)	:TRIGger:SPI:SOURce:FRAMe? (see page 380)	<value> ::= {CHANnel<n> | EXTernal} for the DSO models <value> ::= {CHANnel<n> | DIGital0,..,DIGital15} for the MSO models <n> ::= 1-2 or 1-4 in NR1 format

:TRIGger:SPI:CLOCk:SLOPe

N (see page 530)

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 313
- ":TRIGger:SPI:CLOCk:TIMeout" on page 374
- ":TRIGger:SPI:SOURce:CLOCk" on page 378

:TRIGger:SPI:CLOCk:TIMeout

N (see page 530)

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?
```

The :TRIGger:SPI:CLOCk:TIMeout? query returns current SPI clock timeout setting.

Return Format

```
<time value><NL>
```

```
<time_value> ::= time in seconds in NR1 format
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:SPI:CLOCk:SLOPe" on page 373
- ":TRIGger:SPI:SOURce:CLOCk" on page 378
- ":TRIGger:SPI:FRAMing" on page 375

:TRIGger:SPI:FRAMing

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:SPI:CLOCk:TIMeout" on page 374
- ":TRIGger:SPI:SOURce:FRAMe" on page 380

:TRIGger:SPI:PATTern:DATA

N (see page 530)

Command Syntax

```
:TRIGger:SPI:PATTern:DATA <value>,<mask>
```

```
<value> ::= integer or <string>
```

```
<mask> ::= integer or <string>
```

```
<string> ::= "0xnnnnnnn" 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?
```

The :TRIGger:SPI:PATTern:DATA? query returns the current settings of the specified SPI data pattern resource.

Return Format

```
<value>, <mask><NL>
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:SPI:PATTern:WIDTh" on page 377
- ":TRIGger:SPI:SOURce:DATA" on page 379

:TRIGger:SPI:PATTern:WIDTh

N (see page 530)

Command Syntax

```
:TRIGger:SPI:PATTern:WIDTh <width>
```

```
<width> ::= integer from 4 to 32 in NR1 format
```

The :TRIGger:SPI:PATTern:WIDTh command sets the width of the SPI data pattern anywhere from 4 bits to 32 bits.

Query Syntax

```
:TRIGger:SPI:PATTern:WIDTh?
```

The :TRIGger:SPI:PATTern: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 313
- ":TRIGger:SPI:PATTern:DATA" on page 376
- ":TRIGger:SPI:SOURce:DATA" on page 379

:TRIGger:SPI:SOURce:CLOCk

N (see page 530)

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 313
- ":TRIGger:SPI:CLOCk:SLOPe" on page 373
- ":TRIGger:SPI:CLOCk:TIMeout" on page 374
- ":TRIGger:SPI:SOURce:FRAMe" on page 380
- ":TRIGger:SPI:SOURce:DATA" on page 379

:TRIGger:SPI:SOURce:DATA

N (see page 530)

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 313
- ":TRIGger:SPI:SOURce:CLOCk" on page 378
- ":TRIGger:SPI:SOURce:FRAMe" on page 380
- ":TRIGger:SPI:PATTern:DATA" on page 376
- ":TRIGger:SPI:PATTern:WIDTh" on page 377

:TRIGger:SPI:SOURce:FRAMe

N (see page 530)

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 313
- ":TRIGger:SPI:SOURce:CLOCk" on page 378
- ":TRIGger:SPI:SOURce:DATA" on page 379
- ":TRIGger:SPI:FRAMing" on page 375

:TRIGger:TV Commands

Table 64 :TRIGger:TV Commands Summary

Command	Query	Options and Query Returns
:TRIGger:TV:LINE <line number> (see page 382)	:TRIGger:TV:LINE? (see page 382)	<line number> ::= integer in NR1 format
:TRIGger:TV:MODE <tv mode> (see page 383)	:TRIGger:TV:MODE? (see page 383)	<tv mode> ::= {FIEld1 | FIEld2 | AFIelds | ALINes | LINE | VERTical | LFIeld1 | LFIeld2 | LALTernate | LVERtical}
:TRIGger:TV:POLarity <polarity> (see page 384)	:TRIGger:TV:POLarity? (see page 384)	<polarity> ::= {POSitive | NEGative}
:TRIGger:TV:SOURce <source> (see page 385)	:TRIGger:TV:SOURce? (see page 385)	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 integer in NR1 format
:TRIGger:TV:STANdard <standard> (see page 386)	:TRIGger:TV:STANdard? (see page 386)	<standard> ::= {GENeric | NTSC | PALM | PAL | SECam | {P480L60HZ | P480} | {P720L60HZ | P720} | {P1080L24HZ | P1080} | P1080L25HZ | {I1080L50HZ | I1080} | I1080L60HZ}

:TRIGger:TV:LINE

N (see page 530)

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 65 TV Trigger Line Number Limits

TV Standard	Mode				
	LINE	LFIeld1	LFIeld2	LALTernate	VERTical
NTSC		1 to 263	1 to 262	1 to 262	
PAL		1 to 313	314 to 625	1 to 312	
PAL-M		1 to 263	264 to 525	1 to 262	
SECAM		1 to 313	314 to 625	1 to 312	
GENERIC		1 to 1024	1 to 1024		1 to 1024
P480L60HZ	1 to 525				
P720L60HZ	1 to 750				
P1080L24HZ	1 to 1125				
P1080L25HZ	1 to 1125				
I1080L50HZ	1 to 1125				
I1080L60HZ	1 to 1125				

Query Syntax

```
:TRIGger:TV:LINE?
```

The :TRIGger:TV:LINE? query returns the current TV trigger line number setting.

Return Format

```
<line_number><NL>
```

```
<line_number>::= integer in NR1 format
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:TV:STANdard" on page 386
- ":TRIGger:TV:MODE" on page 383

:TRIGger:TV:MODE

N (see page 530)

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:

<mode>	Old Forms Accepted
FIEld1	F1
FIEld2	F2
AFIelds	ALLFields, ALLFLDS
ALINes	ALLLines
LFIeld1	LINEF1, LINEFIELD1
LFIeld2	LINEF2, LINEFIELD2
LALTernate	LINEAlt
LVERtical	LINEVert

Query Syntax

```
:TRIGger:TV:MODE?
```

The :TRIGger:TV:MODE? query returns the TV trigger mode.

Return Format

```
<value><NL>

<value> ::= {FIE1 | FIE2 | AFI | ALIN | LINE | VERT | LFI1 | LFI2
            | LALT | LVER}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:TV:STANdard" on page 386
- ":TRIGger:MODE" on page 319

:TRIGger:TV:POLarity

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:TV:SOURce" on page 385

:TRIGger:TV:SOURce

N (see page 530)

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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:TV:POLarity" on page 384

Example Code

- "Example Code" on page 346

:TRIGger:TV:STANdard

N (see page 530)

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:UART Commands

Table 66 :TRIGger:UART Commands Summary

Command	Query	Options and Query Returns
:TRIGger:UART:BAUDrate <baudrate> (see page 389)	:TRIGger:UART:BAUDrate? (see page 389)	<baudrate> ::= {1200 | 1800 | 2000 | 2400 | 3600 | 4800 | 7200 | 9600 | 14400 | 15200 | 19200 | 28800 | 38400 | 56000 | 57600 | 76800 | 115200 | 128000 | 230400 | 460800 | 921600 | 1382400 | 1843200 | 2764800}
:TRIGger:UART:BITorder <bitorder> (see page 390)	:TRIGger:UART:BITorder? (see page 390)	<bitorder> ::= {LSBFirst | MSBFirst}
:TRIGger:UART:BURSt <value> (see page 391)	:TRIGger:UART:BURSt? (see page 391)	<value> ::= {OFF | 1 to 4096 in NR1 format}
:TRIGger:UART:DATA <value> (see page 392)	:TRIGger:UART:DATA? (see page 392)	<value> ::= 8-bit integer in decimal or <nondecimal> from 0-255 (0x00-0xff) <nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal <nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
:TRIGger:UART:IDLE <time_value> (see page 393)	:TRIGger:UART:IDLE? (see page 393)	<time_value> ::= time from 10 us to 10 s in NR3 format
:TRIGger:UART:PARity <parity> (see page 394)	:TRIGger:UART:PARity? (see page 394)	<parity> ::= {EVEN | ODD | NONE}
:TRIGger:UART:POLarity <polarity> (see page 395)	:TRIGger:UART:POLarity? (see page 395)	<polarity> ::= {HIGH | LOW}
:TRIGger:UART:QUALifier <value> (see page 396)	:TRIGger:UART:QUALifier? (see page 396)	<value> ::= {EQUal | NOTequal | GREaterthan | LESSthan}
:TRIGger:UART:SOURce:RX <source> (see page 397)	:TRIGger:UART:SOURce:RX? (see page 397)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format

Table 66 :TRIGger:UART Commands Summary (continued)

Command	Query	Options and Query Returns
:TRIGger:UART:SOURce:TX <source> (see page 398)	:TRIGger:UART:SOURce:TX? (see page 398)	<source> ::= {CHANnel<n> | EXTernal} for DSO models <source> ::= {CHANnel<n> | DIGital0,..,DIGital15} for MSO models <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:UART:TYPE <value> (see page 399)	:TRIGger:UART:TYPE? (see page 399)	<value> ::= {RSTArt | RSTop | RDATa | RD1 | RD0 | RDX | PARityerror | TSTArt | TSTOp | TDATa | TD1 | TD0 | TDX}
:TRIGger:UART:WIDTh <width> (see page 400)	:TRIGger:UART:WIDTh? (see page 400)	<width> ::= {5 | 6 | 7 | 8 | 9}

:TRIGger:UART:BAUDrate

N (see page 530)

Command Syntax

```
:TRIGger:UART:BAUDrate <baudrate>

<baudrate> ::= integer in NR1 format

<baudrate> ::= {1200 | 1800 | 2000 | 2400 | 3600 | 4800 | 7200 | 9600
               | 14400 | 15200 | 19200 | 28800 | 38400 | 56000 | 57600
               | 76800 | 115200 | 128000 | 230400 | 460800 | 921600
               | 1382400 | 1843200 | 2764800}
```

The :TRIGger:UART:BAUDrate command selects the bit rate (in bps) for the serial decoder and/or trigger when in UART mode.

If the baud rate you select does not match the system baud rate, false triggers may occur.

Query Syntax

```
:TRIGger:UART:BAUDrate?
```

The :TRIGger:UART:BAUDrate? query returns the current UART baud rate setting.

Return Format

```
<baudrate><NL>

<baudrate> ::= integer = {600 to 1000000 | 1382000 | 1843000 | 2765000}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:BITorder

N (see page 530)

Command Syntax

```
:TRIGger:UART:BITorder <bitorder>
```

```
<bitorder> ::= {LSBFirst | MSBFirst}
```

The :TRIGger:UART:BITorder command specifies the order of transmission used by the physical Tx and Rx input signals for the serial decoder and/or trigger when in UART mode. LSBFirst sets the least significant bit of each message "byte" as transmitted first. MSBFirst sets the most significant bit as transmitted first.

Query Syntax

```
:TRIGger:UART:BITorder?
```

The :TRIGger:UART:BITorder? query returns the current UART bit order setting.

Return Format

```
<bitorder><NL>
```

```
<bitorder> ::= {LSBF | MSBF}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399
- ":TRIGger:UART:SOURce:RX" on page 397
- ":TRIGger:UART:SOURce:TX" on page 398

:TRIGger:UART:BURSt

N (see page 530)

Command Syntax

```
:TRIGger:UART:BURSt <value>

<value> ::= {OFF | 1 to 4096 in NR1 format}
```

The :TRIGger:UART:BURSt command selects the burst value (Nth occurrence of trigger event after idle period) in the range 1 to 4096 or OFF, for the trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:BURSt?
```

The :TRIGger:UART:BURSt? query returns the current UART trigger burst value.

Return Format

```
<value><NL>

<value> ::= {OFF | 1 to 4096 in NR1 format}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:IDLE" on page 393
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:DATA

N (see page 530)

Command Syntax

```
:TRIGger:UART:DATA <value>

<value> ::= 8-bit integer in decimal or <nondecimal> from 0-255
            (0x00-0xff)

<nondecimal> ::= #Hnn where n ::= {0,..,9 | A,..,F} for hexadecimal

<nondecimal> ::= #Bnn...n where n ::= {0 | 1} for binary
```

The :TRIGger:UART:DATA command selects the data byte value (0x00 to 0xFF) for the trigger QUALifier when in UART mode. The data value is used when one of the RD or TD trigger types is selected.

Query Syntax

```
:TRIGger:UART:DATA?
```

The :TRIGger:UART:DATA? query returns the current UART trigger data value.

Return Format

```
<value><NL>

<value> ::= 8-bit integer in decimal from 0-255
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:IDLE

N (see page 530)

Command Syntax

```
:TRIGger:UART:IDLE <time_value>
```

```
<time_value> ::= time from 10 us to 10 s in NR3 format
```

The :TRIGger:UART:IDLE command selects the value of the idle period for burst trigger in the range from 1 us to 10 s when in UART mode.

Query Syntax

```
:TRIGger:UART:IDLE?
```

The :TRIGger:UART:IDLE? query returns the current UART trigger idle period time.

Return Format

```
<time_value><NL>
```

```
<time_value> ::= time from 10 us to 10 s in NR3 format
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:BURSt" on page 391
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:PARity

N (see page 530)

Command Syntax

```
:TRIGger:UART:PARity <parity>
```

```
<parity> ::= {EVEN | ODD | NONE}
```

The :TRIGger:UART:PARity command selects the parity to be used with each message "byte" for the serial decoder and/or trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:PARity?
```

The :TRIGger:UART:PARity? query returns the current UART parity setting.

Return Format

```
<parity><NL>
```

```
<parity> ::= {EVEN | ODD | NONE}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:POLarity

N (see page 530)

Command Syntax

```
:TRIGger:UART:POLarity <polarity>
```

```
<polarity> ::= {HIGH | LOW}
```

The :TRIGger:UART:POLarity command selects the polarity as idle low or idle high for the serial decoder and/or trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:POLarity?
```

The :TRIGger:UART:POLarity? query returns the current UART polarity setting.

Return Format

```
<polarity><NL>
```

```
<polarity> ::= {HIGH | LOW}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:QUALifier

N (see page 530)

Command Syntax

```
:TRIGger:UART:QUALifier <value>
```

```
<value> ::= {EQUal | NOTequal | GREaterthan | LESSthan}
```

The :TRIGger:UART:QUALifier command selects the data qualifier when :TYPE is set to RDATa, RD1, RD0, RDX, TDATa, TD1, TD0, or TDX for the trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:QUALifier?
```

The :TRIGger:UART:QUALifier? query returns the current UART trigger qualifier.

Return Format

```
<value><NL>
```

```
<value> ::= {EQU | NOT | GRE | LESS}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:TRIGger:UART:SOURce:RX

N (see page 530)

Command Syntax

```
:TRIGger:UART:SOURce:RX <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:UART:SOURce:RX command controls which signal is used as the Rx source by the serial decoder and/or trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:SOURce:RX?
```

The :TRIGger:UART:SOURce:RX? query returns the current source for the UART Rx signal.

Return Format

```
<source><NL>
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399
- ":TRIGger:UART:BITorder" on page 390

:TRIGger:UART:SOURce:TX

N (see page 530)

Command Syntax

```
:TRIGger:UART:SOURce:TX <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:UART:SOURce:TX command controls which signal is used as the Tx source by the serial decoder and/or trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:SOURce:TX?
```

The :TRIGger:UART:SOURce:TX? query returns the current source for the UART Tx signal.

Return Format

```
<source><NL>
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399
- ":TRIGger:UART:BITorder" on page 390

:TRIGger:UART:TYPE

N (see page 530)

Command Syntax

```
:TRIGger:UART:TYPE <value>

<value> ::= {RSTArt | RSTop | RDATa | RD1 | RD0 | RDX | PARityerror
             | TSTArt | TSTOp | TDATa | TD1 | TD0 | TDX}
```

The :TRIGger:UART:TYPE command selects the UART trigger type.

When one of the RD or TD types is selected, the :TRIGger:UART:DATA and :TRIGger:UART:QUALifier commands are used to specify the data value and comparison operator.

The RD1, RD0, RDX, TD1, TD0, and TDX types (for triggering on data and alert bit values) are only valid when a 9-bit width has been selected.

Query Syntax

```
:TRIGger:UART:TYPE?
```

The :TRIGger:UART:TYPE? query returns the current UART trigger data value.

Return Format

```
<value><NL>

<value> ::= {RSTA | RST | RDAT | RD1 | RD0 | RDX | PAR | TSTA |
            TSTO | TDAT | TD1 | TD0 | TDX}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:DATA" on page 392
- ":TRIGger:UART:QUALifier" on page 396
- ":TRIGger:UART:WIDTh" on page 400

:TRIGger:UART:WIDTh

N (see page 530)

Command Syntax

```
:TRIGger:UART:WIDTh <width>
```

```
<width> ::= {5 | 6 | 7 | 8 | 9}
```

The :TRIGger:UART:WIDTh command determines the number of bits (5-9) for each message "byte" for the serial decoder and/or trigger when in UART mode.

Query Syntax

```
:TRIGger:UART:WIDTh?
```

The :TRIGger:UART:WIDTh? query returns the current UART width setting.

Return Format

```
<width><NL>
```

```
<width> ::= {5 | 6 | 7 | 8 | 9}
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:UART:TYPE" on page 399

:WAVeform Commands

Provide access to waveform data. See "Introduction to :WAVeform Commands" on page 403.

Table 67 :WAVeform Commands Summary

Command	Query	Options and Query Returns
:WAVeform:BYTeorder <value> (see page 408)	:WAVeform:BYTeorder? (see page 408)	<value> ::= {LSBFirst | MSBFirst}
n/a	:WAVeform:COUNt? (see page 409)	<count> ::= an integer from 1 to 65536 in NR1 format
n/a	:WAVeform:DATA? (see page 410)	<binary block length bytes>, <binary data> For example, to transmit 1000 bytes of data, the syntax would be: #800001000<1000 bytes of data><NL> 8 is the number of digits that follow 00001000 is the number of bytes to be transmitted <1000 bytes of data> is the actual data
:WAVeform:FORMat <value> (see page 412)	:WAVeform:FORMat? (see page 412)	<value> ::= {WORD | BYTE | ASCII}
:WAVeform:POINts <# points> (see page 413)	:WAVeform:POINts? (see page 413)	<# points> ::= {100 | 250 | 500 | 1000 | <points_mode>} if waveform points mode is NORMal <# points> ::= {100 | 250 | 500 | 1000 | 2000 ... 8000000 in 1-2-5 sequence | <points_mode>} if waveform points mode is MAXimum or RAW <points_mode> ::= {NORMal | MAXimum | RAW}
:WAVeform:POINts:MODE <points_mode> (see page 415)	:WAVeform:POINts:MODE? (see page 415)	<points_mode> ::= {NORMal | MAXimum | RAW}

Table 67 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVeform:PREamble? (see page 417)	<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 420)	:WAVeform:SOURce? (see page 420)	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format
:WAVeform:SOURce:SUBSource <subsource> (see page 424)	:WAVeform:SOURce:SUBSource? (see page 424)	<subsource> ::= {{NONE | RX} | TX}
n/a	:WAVeform:TYPE? (see page 425)	<return_mode> ::= {NORM | PEAK | AVER | HRES}
:WAVeform:UNSigned {{0 | OFF} | {1 | ON}} (see page 426)	:WAVeform:UNSigned? (see page 426)	{0 | 1}
:WAVeform:VIEW <view> (see page 427)	:WAVeform:VIEW? (see page 427)	<view> ::= {MAIN}
n/a	:WAVeform:XINCrement? (see page 428)	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVeform:XORigin? (see page 429)	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVeform:XREFerence? (see page 430)	<return_value> ::= 0 (x-reference value in the current preamble in NR1 format)

Table 67 :WAVeform Commands Summary (continued)

Command	Query	Options and Query Returns
n/a	:WAVeform:YINCrement? (see page 431)	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVeform:YORigin? (see page 432)	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVeform:YREFerence? (see page 433)	<return_value> ::= y-reference value in the current preamble in NR1 format

Introduction to :WAVeform Commands

The WAVeform subsystem is used to transfer data to a controller from the oscilloscope waveform memories. The queries in this subsystem will only operate when the channel selected by :WAVeform:SOURce is on.

Waveform Data and Preamble

The waveform record is actually contained in two portions: the preamble and waveform data. The waveform record must be read from the oscilloscope by the controller using two separate commands, :WAVeform:DATA (see page 410) and :WAVeform:PREamble (see page 417). 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 132): NORMal, AVERage, PEAK, and HRESolution. When the data is acquired using the :DIGitize command (see page 96) or :RUN command (see page 116), 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 410) 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 130).

Helpful Hints:

The number of points transferred to the computer is controlled using the :WAVeform:POINts command (see page 413). 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. 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 413). 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 127). 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 413). 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 413). 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 132), the value returned by the :WAVeform:XINCrement query (see page 428) 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:

voltage = [(data value - yreference) * yincrement] + yorigin

If the :WAVeform:FORMat data format is ASCii (see page 412), the data values are converted internally and sent as floating point values separated by commas.

In converting a data value to time, the time value of a data point can be determined by the position of the data point. For example, the fourth data point sent with :WAVeform:XORigin = 16 ns, :WAVeform:XREFerence = 0, and :WAVeform:XINCrement = 2 ns, can be calculated using the following formula:

time = [(data point number - xreference) * xincrement] + xorigin

This would result in the following calculation for time bucket 3:

time = [(3 - 0) * 2 ns] + 16 ns = 22 ns

In :ACQuire:TYPE PEAK mode (see page 132), because data is acquired in max-min pairs, modify the previous time formula to the following:

time=[(data pair number - xreference) * xincrement * 2] + xorigin

Data Format for Transfer

There are three formats for transferring waveform data over the interface: BYTE, WORD and ASCii (see ":WAVeform:FORMat" on page 412). BYTE, WORD and ASCii formatted waveform records are transmitted using the arbitrary block program data format specified in IEEE 488.2.

When you use the block data format, the ASCII character string "#8<DD...D>" is sent prior to sending the actual data. The 8 indicates how many Ds follow. The Ds are ASCII numbers that indicate how many data bytes follow.

For example, if 1000 points will be transferred, and the WORD format was specified, the block header "#800001000" would be sent. The 8 indicates that eight length bytes follow, and 00001000 indicates that 1000 binary data bytes follow.

Use the :WAVeform:UNSigned command (see page 426) to control whether data values are sent as unsigned or signed integers. This command can be used to match the instrument's internal data type to the data type used by the programming language. This command has no effect if the data format is ASCii.

Data Format for Transfer - ASCii format

The ASCii format (see ":WAVeform:FORMat" on page 412) provides access to the waveform data as real Y-axis values without using Y origin, Y reference, and Y increment to convert the binary data. Values are transferred as ASCii digits in floating point format separated by commas. In ASCii format, holes are represented by the value 9.9e+37. The setting of :WAVeform:BYTeorder (see page 408) and :WAVeform:UNSigned (see page 426) have no effect when the format is ASCii.

Data Format for Transfer - WORD format

WORD format (see ":WAVeform:FORMat" on page 412) 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 413). 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 408) 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 412) 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 408) has no effect when the data format is BYTE.

Reporting the Setup

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

```
:WAV:UNS 1;VIEW MAIN;BYT MSBF;FORM BYTE;POIN +1000;SOUR CHAN1;SOUR:SUBS
NONE
```

:WAVeform:BYTeorder

C (see page 530)

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 403
- ":WAVeform:DATA" on page 410
- ":WAVeform:FORMat" on page 412
- ":WAVeform:PREamble" on page 417

Example Code

- "Example Code" on page 420
- "Example Code" on page 418

:WAVeform:COUNt

C (see page 530)

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 403
- ":ACQuire:COUNt" on page 127
- ":ACQuire:TYPE" on page 132

:WAVeform:DATA

C (see page 530)

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 403
- ":WAVeform:UNSigned" on page 426
- ":WAVeform:BYTeorder" on page 408
- ":WAVeform:FORMat" on page 412
- ":WAVeform:POINts" on page 413
- ":WAVeform:PREamble" on page 417
- ":WAVeform:SOURce" on page 420
- ":WAVeform:TYPE" on page 425

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 578

:WAVeform:FORMat

C (see page 530)

Command Syntax

```
:WAVeform:FORMat <value>
```

```
<value> ::= {WORD | BYTE | ASCii}
```

The :WAVeform:FORMat command sets the data transmission mode for waveform data points. This command controls how the data is formatted when sent from the oscilloscope.

- ASCii formatted data converts the internal integer data values to real Y-axis values. Values are transferred as ASCii digits in floating point notation, separated by commas.

  ASCII formatted data is transferred ASCii text.
- WORD formatted data transfers 16-bit data as two bytes. The :WAVeform:BYTeorder command can be used to specify whether the upper or lower byte is transmitted first. The default (no command sent) is that the upper byte transmitted first.
- BYTE formatted data is transferred as 8-bit bytes.

Query Syntax

```
:WAVeform:FORMat?
```

The :WAVeform:FORMat query returns the current output format for the transfer of waveform data.

Return Format

```
<value><NL>
```

```
<value> ::= {WORD | BYTE | ASC}
```

See Also

- ["Introduction to :WAVeform Commands"](#) on page 403
- [":WAVeform:BYTeorder"](#) on page 408
- [":WAVeform:DATA"](#) on page 410
- [":WAVeform:PREamble"](#) on page 417

Example Code

- ["Example Code"](#) on page 420

:WAVeform:POINts

C (see page 530)

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 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 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 415) 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.

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 415) for more information).

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 403
- ":ACQuire:POINts" on page 130
- ":WAVeform:DATA" on page 410
- ":WAVeform:SOURce" on page 420
- ":WAVeform:VIEW" on page 427
- ":WAVeform:PREamble" on page 417
- ":WAVeform:POINts:MODE" on page 415

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 578

:WAVeform:POINts:MODE

N (see page 530)

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 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 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 (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 displayed is changing, the data being retrieved can switch between the measurement and raw acquisition records.

Considerations for MAXimum or RAW data retrieval

- The instrument must be stopped (see the :STOP command (see page 120) or the :DIGitize command (see page 96) 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>

<points_mode> ::= {NORMal | MAXimum | RAW}
```

See Also

- "Introduction to :WAVeform Commands" on page 403
- ":ACQuire:POINts" on page 130
- ":WAVeform:DATA" on page 410
- ":WAVeform:VIEW" on page 427
- ":WAVeform:PREamble" on page 417
- ":WAVeform:POINts" on page 413
- ":TIMebase:MODE" on page 304
- ":ACQuire:TYPE" on page 132
- ":ACQuire:COUNt" on page 127

:WAVeform:PREamble

C (see page 530)

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

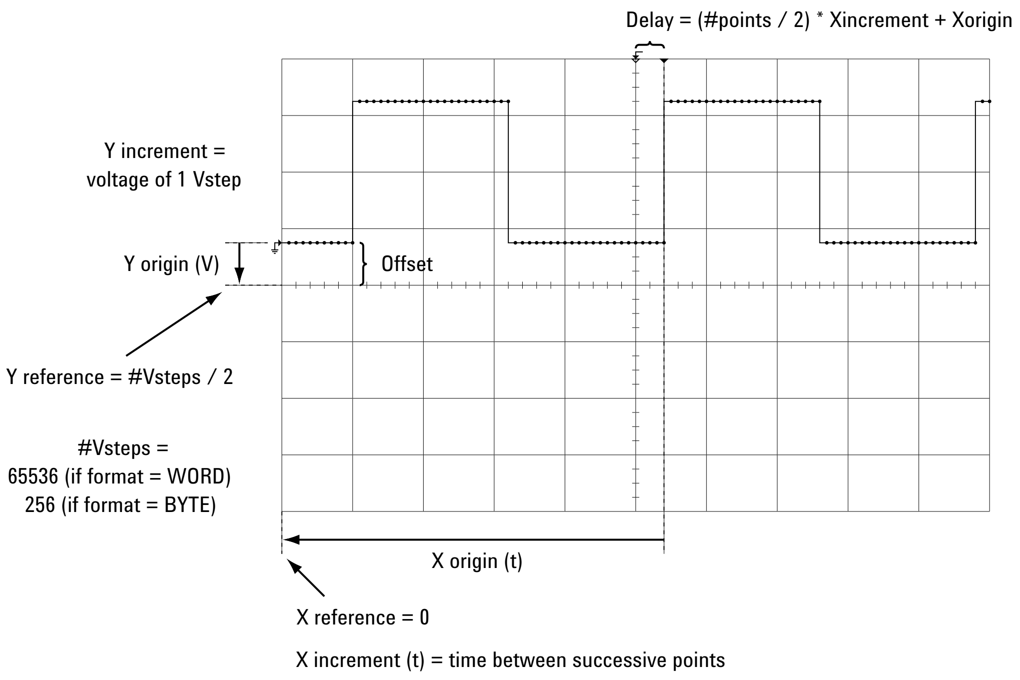
```
<preamble_block><NL>

<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).
```


Delay = (#points / 2) * Xincrement + Xorigin
Y increment = voltage of 1 Vstep
Y origin (V)
Offset
Y reference = #Vsteps / 2
#Vsteps = 65536 (if format = WORD) 256 (if format = BYTE)
X origin (t)
X reference = 0
X increment (t) = time between successive points

See Also

- "Introduction to :WAVeform Commands" on page 403
- ":ACQuire:COUNt" on page 127
- ":ACQuire:POINts" on page 130
- ":ACQuire:TYPE" on page 132
- ":DIGitize" on page 96
- ":WAVeform:COUNt" on page 409
- ":WAVeform:DATA" on page 410
- ":WAVeform:FORMat" on page 412
- ":WAVeform:POINts" on page 413
- ":WAVeform:TYPE" on page 425
- ":WAVeform:XINCrement" on page 428
- ":WAVeform:XORigin" on page 429
- ":WAVeform:XREFerence" on page 430
- ":WAVeform:YINCrement" on page 431
- ":WAVeform:YORigin" on page 432
- ":WAVeform:YREFerence" on page 433

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 - 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)
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:WAVeform:SOURce

C (see page 530)

Command Syntax

```
:WAVeform:SOURce <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 :WAVeform:SOURce command selects the analog channel or function to be used as the source for the :WAVeform commands.

Function capabilities include add, subtract, multiply; integrate, differentiate, and FFT (Fast Fourier Transform) operations.

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}

<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 403
- ":DIGitize" on page 96
- ":WAVeform:FORMat" on page 412
- ":WAVeform:BYTeorder" on page 408
- ":WAVeform:DATA" on page 410
- ":WAVeform:PREamble" on page 417

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) + _
            " V" + vbCrLf
strOutput = strOutput + "Offset = " + _
            FormatNumber((lngVSteps / 2 - lngYReference) * _
            sngYIncrement + sngYOrigin) + " V" + vbCrLf
strOutput = strOutput + "Sec/Div = " + _
            FormatNumber(lngPoints * dblXIncrement / 10 * _
            1000000) + " us" + vbCrLf
strOutput = strOutput + "Delay = " + _
            FormatNumber(((lngPoints / 2 - lngXReference) * _
            dblXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf

' QUERY_WAVE_DATA - Outputs waveform data that is stored in a buffer.

' Query the oscilloscope for the waveform data.
myScope.WriteString ":WAV:DATA?"

' READ_WAVE_DATA - The wave data consists of two parts: the header,
' and the actual waveform data followed by a new line (NL) character.
' The query data has the following format:
'
'    <header><waveform_data><NL>
'
' Where:
'    <header> = #800001000 (This is an example header)
' The "#8" may be stripped off of the header and the remaining
' numbers are the size, in bytes, of the waveform data block.  The
' size can vary depending on the number of points acquired for the
' waveform.  You can then read that number of bytes from the
' oscilloscope and the terminating NL character.
'
Dim lngI As Long
Dim lngDataValue As Long

' Unsigned integer bytes.
varQueryResult = myScope.ReadIEEEBlock(BinaryType_UI1)

For lngI = 0 To UBound(varQueryResult) _
```

```
      Step (UBound(varQueryResult) / 20)   ' 20 points.
    If intBytesPerData = 2 Then
      lngDataValue = varQueryResult(lngI) * 256 _
          + varQueryResult(lngI + 1)   ' 16-bit value.
    Else
      lngDataValue = varQueryResult(lngI)   ' 8-bit value.
    End If
    strOutput = strOutput + "Data point " + _
      CStr(lngI / intBytesPerData) + ", " + _
      FormatNumber((lngDataValue - lngYReference) _
          * sngYIncrement + sngYOrigin) + " V, " + _
      FormatNumber(((lngI / intBytesPerData - lngXReference) _
          * sngXIncrement + dblXOrigin) * 1000000) + " us" + vbCrLf
  Next lngI
  MsgBox "Waveform data:" + vbCrLf + strOutput
```

Example program from the start: "VISA COM Example in Visual Basic" on page 578

:WAVeform:SOURce:SUBSource

C (see page 530)

Command Syntax

```
:WAVeform:SOURce:SUBSource <subsource>
```

```
<subsource> ::= {{NONE | RX} | TX}
```

If the :WAVeform:SOURce is SBUS (serial decode), more than one data set may be available, and this command lets you choose from the available data sets.

Currently, only UART serial decode lets you get "TX" data. The default, NONE, specifies "RX" data. (RX is an alias for NONE.)

If the :WAVeform:SOURce is not SBUS, or the :SBUS:MODE is not UART, the only valid subsource is NONE.

Query Syntax

```
:WAVeform:SOURce:SUBSource?
```

The :WAVeform:SOURce:SUBSource? query returns the current waveform subsource setting.

Return Format

```
<subsource><NL>
```

```
<subsource> ::= {NONE | TX}
```

See Also

- "Introduction to :WAVeform Commands" on page 403
- ":WAVeform:SOURce" on page 420

:WAVeform:TYPE

C (see page 530)

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}
```

See Also

- "Introduction to :WAVeform Commands" on page 403
- ":ACQuire:TYPE" on page 132
- ":WAVeform:DATA" on page 410
- ":WAVeform:PREamble" on page 417
- ":WAVeform:SOURce" on page 420

:WAVeform:UNSigned

C (see page 530)

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.

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 403
- ":WAVeform:SOURce" on page 420

:WAVeform:VIEW

C (see page 530)

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 403
- ":WAVeform:POINts" on page 413

:WAVeform:XINCrement

(see page 530)

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 403
- ":WAVeform:PREamble" on page 417

Example Code

- "Example Code" on page 418

:WAVeform:XORigin

C (see page 530)

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 403
- ":WAVeform:PREamble" on page 417
- ":WAVeform:XREFerence" on page 430

Example Code

- "Example Code" on page 418

:WAVeform:XREFerence

C (see page 530)

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 403
- ":WAVeform:PREamble" on page 417
- ":WAVeform:XORigin" on page 429

Example Code

- "Example Code" on page 418

:WAVeform:YINCrement

C (see page 530)

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.

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 403
- ":WAVeform:PREamble" on page 417

Example Code

- "Example Code" on page 418

:WAVeform:YORigin

C (see page 530)

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 403
- ":WAVeform:PREamble" on page 417
- ":WAVeform:YREFerence" on page 433

Example Code

- "Example Code" on page 418

:WAVeform:YREFerence

C (see page 530)

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 403
- ":WAVeform:PREamble" on page 417
- ":WAVeform:YORigin" on page 432

Example Code

- "Example Code" on page 418

4
Commands A-Z

A 435
B 436
C 436
D 438
E 439
F 440
G 441
H 441
I 441
L 442
M 443
N 444
O 444
P 445
Q 446
R 446
S 447
T 450
U 453
V 454
W 455
X 455
Y 456

A

- AALias, ":ACQuire:AALias" on page 125
- ":ACQuire:AALias" on page 125
- ":ACQuire:COMPlete" on page 126
- ":ACQuire:COUNt" on page 127
- ":ACQuire:DAALias" on page 128
- ":ACQuire:MODE" on page 129

- ":ACQuire:POINts" on page 130
- ":ACQuire:SRATe" on page 131
- ":ACQuire:TYPE" on page 132
- ADDRess, ":TRIGger:IIC:PATTern:ADDRess" on page 356
- ":AER (Arm Event Register)" on page 89
- APRinter, ":HARDcopy:APRinter" on page 196
- AREA Commands:
  - ":HARDcopy:AREA" on page 195
  - ":SAVE:IMAGe:AREA" on page 266
- ASIZe, ":SBUS:IIC:ASIZe" on page 284
- ":AUToscale" on page 90
  - ":AUToscale:AMODE" on page 92
  - ":AUToscale:CHANnels" on page 93

B

- BASE, ":SBUS:UART:BASE" on page 288
- BAUDrate Commands:
  - ":TRIGger:CAN:SIGNal:BAUDrate" on page 331
  - ":TRIGger:LIN:SIGNal:BAUDrate" on page 367
  - ":TRIGger:UART:BAUDrate" on page 389
- BITorder, ":TRIGger:UART:BITorder" on page 390
- ":BLANk" on page 94
- BURSt, ":TRIGger:UART:BURSt" on page 391
- BWLimit Commands:
  - ":CHANnel<n>:BWLimit" on page 145
  - ":EXTernal:BWLimit" on page 173
- BYTeorder, ":WAVeform:BYTeorder" on page 408

C

- ":CALibrate:DATE" on page 135
- ":CALibrate:LABel" on page 136
- ":CALibrate:STARt" on page 137
- ":CALibrate:STATus" on page 138
- ":CALibrate:SWITch" on page 139
- ":CALibrate:TEMPerature" on page 140
- ":CALibrate:TIME" on page 141
- CAN Commands:
  - ":SBUS:CAN:COUNt:ERRor" on page 278

  - ":SBUS:CAN:COUNt:OVERload" on page 279
  - ":SBUS:CAN:COUNt:RESet" on page 280
  - ":SBUS:CAN:COUNt:TOTal" on page 281
  - ":SBUS:CAN:COUNt:UTILization" on page 282
  - ":TRIGger:CAN Commands" on page 324
- ":CDISplay" on page 95
- CENTer, ":FUNCtion:CENTer" on page 183
- ":CHANnel:LABel" on page 462
- ":CHANnel2:SKEW" on page 463
- ":CHANnel<n>:BWLimit" on page 145
- ":CHANnel<n>:COUPling" on page 146
- ":CHANnel<n>:DISPlay" on page 147
- ":CHANnel<n>:IMPedance" on page 148
- ":CHANnel<n>:INPut" on page 464
- ":CHANnel<n>:INVert" on page 149
- ":CHANnel<n>:LABel" on page 150
- ":CHANnel<n>:OFFSet" on page 151
- ":CHANnel<n>:PMODe" on page 465
- ":CHANnel<n>:PROBe" on page 152
- ":CHANnel<n>:PROBe:ID" on page 153
- ":CHANnel<n>:PROBe:SKEW" on page 154
- ":CHANnel<n>:PROBe:STYPe" on page 155
- ":CHANnel<n>:PROTection" on page 156
- ":CHANnel<n>:RANGe" on page 157
- ":CHANnel<n>:SCALe" on page 158
- ":CHANnel<n>:UNITs" on page 159
- ":CHANnel<n>:VERNier" on page 160
- CLEar Commands:
  - ":DISPlay:CLEar" on page 163
  - ":MEASure:CLEar" on page 220
- CLOCk Commands:
  - ":TRIGger:IIC:SOURce:CLOCk" on page 359
  - ":TRIGger:SPI:CLOCk:SLOPe" on page 373
  - ":TRIGger:SPI:CLOCk:TIMeout" on page 374
  - ":TRIGger:SPI:SOURce:CLOCk" on page 378

- "*CLS (Clear Status)" on page 65
- COMPlete, ":ACQuire:COMPlete" on page 126
- CONNect, ":DISPlay:CONNect" on page 466
- COUNt Commands:
  - ":ACQuire:COUNt" on page 127
  - ":SBUS:CAN:COUNt:ERRor" on page 278
  - ":SBUS:CAN:COUNt:OVERload" on page 279
  - ":SBUS:CAN:COUNt:RESet" on page 280
  - ":SBUS:CAN:COUNt:TOTal" on page 281
  - ":SBUS:CAN:COUNt:UTILization" on page 282
  - ":SBUS:UART:COUNt:ERRor" on page 289
  - ":SBUS:UART:COUNt:RESet" on page 290
  - ":SBUS:UART:COUNt:RXFRames" on page 291
  - ":SBUS:UART:COUNt:TXFRames" on page 292
  - ":WAVeform:COUNt" on page 409
- COUNter, ":MEASure:COUNter" on page 221
- COUPling Commands:
  - ":CHANnel<n>:COUPling" on page 146
  - ":TRIGger[:EDGE]:COUPling" on page 342

D

- DAALias, ":ACQuire:DAALias" on page 128
- DATA Commands:
  - ":DISPlay:DATA" on page 164
  - ":TRIGger:CAN:PATTern:DATA" on page 326
  - ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
  - ":TRIGger:IIC:PATTern:DATA" on page 357
  - ":TRIGger:IIC:PATTern:DATa2" on page 358
  - ":TRIGger:IIC:SOURce:DATA" on page 360
  - ":TRIGger:SPI:PATTern:DATA" on page 376
  - ":TRIGger:SPI:SOURce:DATA" on page 379
  - ":TRIGger:UART:DATA" on page 392
  - ":WAVeform:DATA" on page 410
- DATE Commands:
  - ":CALibrate:DATE" on page 135
  - ":SYSTem:DATE" on page 295
- DEFine, ":MEASure:DEFine" on page 222

- DELay Commands:
  - ":MEASure:DELay" on page 225
  - ":TIMebase:DELay" on page 494
- DESTination, ":HARDcopy:DESTination" on page 471
- DEVice, ":HARDcopy:DEVice" on page 472
- ":DIGitize" on page 96
- DISPlay Commands:
  - ":CHANnel<n>:DISPlay" on page 147
  - ":FUNCtion:DISPlay" on page 184
  - ":SBUS:DISPlay" on page 283
- ":DISPlay:CLEar" on page 163
- ":DISPlay:CONNect" on page 466
- ":DISPlay:DATA" on page 164
- ":DISPlay:LABel" on page 166
- ":DISPlay:LABList" on page 167
- ":DISPlay:PERSistence" on page 168
- ":DISPlay:SOURce" on page 169
- ":DISPlay:VECTors" on page 170
- DSP, ":SYSTem:DSP" on page 296
- DURation, ":TRIGger:DURation Commands" on page 335
- DUTYcycle, ":MEASure:DUTYcycle" on page 227

E

- EDGE, ":TRIGger[:EDGE] Commands" on page 341
- ":ERASe" on page 467
- ERRor Commands:
  - ":SBUS:CAN:COUNt:ERRor" on page 278
  - ":SBUS:UART:COUNt:ERRor" on page 289
  - ":SYSTem:ERRor" on page 297
- "*ESE (Standard Event Status Enable)" on page 66
- "*ESR (Standard Event Status Register)" on page 68
- ":EXTernal:BWLimit" on page 173
- ":EXTernal:IMPedance" on page 174
- ":EXTernal:INPut" on page 468
- ":EXTernal:PMODe" on page 469
- ":EXTernal:PROBe" on page 175
- ":EXTernal:PROBe:ID" on page 176

- ":EXTernal:PROBe:STYPe" on page 177
- ":EXTernal:PROTection" on page 178
- ":EXTernal:RANGe" on page 179
- ":EXTernal:UNITs" on page 180

F

- FACTors Commands:
  - ":HARDcopy:FACTors" on page 197
  - ":SAVE:IMAGe:FACTors" on page 267
- FALLtime, ":MEASure:FALLtime" on page 228
- FFEed, ":HARDcopy:FFEed" on page 198
- FILename Commands:
  - ":HARDcopy:FILename" on page 473
  - ":RECall:FILename" on page 258
  - ":SAVE:FILename" on page 264
- FORMat Commands:
  - ":HARDcopy:FORMat" on page 474
  - ":SAVE:IMAGe:FORMat" on page 268
  - ":SAVE:WAVeform:FORMat" on page 274
  - ":WAVeform:FORMat" on page 412
- FRAMe, ":TRIGger:SPI:SOURce:FRAMe" on page 380
- FRAMing Commands:
  - ":SBUS:UART:FRAMing" on page 293
  - ":TRIGger:SPI:FRAMing" on page 375
- FREQuency, ":MEASure:FREQuency" on page 229
- ":FUNCtion:CENTer" on page 183
- ":FUNCtion:DISPlay" on page 184
- ":FUNCtion:OFFSet" on page 185
- ":FUNCtion:OPERation" on page 186
- ":FUNCtion:RANGe" on page 187
- ":FUNCtion:REFerence" on page 188
- ":FUNCtion:SCALe" on page 189
- ":FUNCtion:SOURce" on page 190
- ":FUNCtion:SPAN" on page 191
- ":FUNCtion:VIEW" on page 470
- ":FUNCtion:WINDow" on page 192

G
- GLITch (Pulse Width), ":TRIGger:GLITch Commands" on page 347
- GRAYscale, ":HARDcopy:GRAYscale" on page 475
- GREaterthan Commands:
  - ":TRIGger:DURation:GREaterthan" on page 336
  - ":TRIGger:GLITch:GREaterthan" on page 348

H
- ":HARDcopy:AREA" on page 195
- ":HARDcopy:APRinter" on page 196
- ":HARDcopy:DESTination" on page 471
- ":HARDcopy:DEVice" on page 472
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:FFEed" on page 198
- ":HARDcopy:FILename" on page 473
- ":HARDcopy:FORMat" on page 474
- ":HARDcopy:GRAYscale" on page 475
- ":HARDcopy:IGColors" on page 476
- ":HARDcopy:INKSaver" on page 199
- ":HARDcopy:PALette" on page 200
- ":HARDcopy:PDRiver" on page 477
- ":HARDcopy:PRinter:LIST" on page 201
- ":HARDcopy:STARt" on page 202
- HFReject, ":TRIGger:HFReject" on page 317
- HOLDoff, ":TRIGger:HOLDoff" on page 318

I
- ID Commands:
  - ":TRIGger:CAN:PATTern:ID" on page 328
  - ":TRIGger:CAN:PATTern:ID:MODE" on page 329
- IDLE, ":TRIGger:UART:IDLE" on page 393
- "*IDN (Identification Number)" on page 70
- IIC Commands:
  - ":SBUS:IIC:ASIZe" on page 284
  - ":TRIGger:IIC Commands" on page 355
- IGColors Commands:
  - ":HARDcopy:IGColors" on page 476
  - ":SAVE:IMAGe:INKSaver" on page 269
- IMAGe Commands:

  - ":RECall:IMAGe[:STARt]" on page 259
  - ":SAVE:IMAGe:AREA" on page 266
  - ":SAVE:IMAGe:FACTors" on page 267
  - ":SAVE:IMAGe:FORMat" on page 268
  - ":SAVE:IMAGe:INKSaver" on page 269
  - ":SAVE:IMAGe:PALette" on page 270
  - ":SAVE:IMAGe[:STARt]" on page 265
- IMPedance Commands:
  - ":CHANnel<n>:IMPedance" on page 148
  - ":EXTernal:IMPedance" on page 174
- INKSaver, ":HARDcopy:INKSaver" on page 199
- INVert, ":CHANnel<n>:INVert" on page 149

L

- LABel Commands:
  - ":CALibrate:LABel" on page 136
  - ":CHANnel:LABel" on page 462
  - ":CHANnel<n>:LABel" on page 150
  - ":DISPlay:LABel" on page 166
- LABList, ":DISPlay:LABList" on page 167
- LENGth Commands:
  - ":SAVE:WAVeform:LENGth" on page 275
  - ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
- LESSthan Commands:
  - ":TRIGger:DURation:LESSthan" on page 337
  - ":TRIGger:GLITch:LESSthan" on page 349
- LEVel Commands:
  - ":TRIGger[:EDGE]:LEVel" on page 343
  - ":TRIGger:GLITch:LEVel" on page 350
- LIN Commands:
  - ":SBUS:LIN:PARity" on page 285
  - ":TRIGger:LIN Commands" on page 364
- LINE, ":TRIGger:TV:LINE" on page 382
- LIST, ":HARDcopy:PRinter:LIST" on page 201
- LOCK, ":SYSTem:LOCK" on page 298
- LOWer, ":MEASure:LOWer" on page 478
- "*LRN (Learn Device Setup)" on page 71

M
- ":MARKer:MODE" on page 205
- ":MARKer:X1Position" on page 206
- ":MARKer:X1Y1source" on page 207
- ":MARKer:X2Position" on page 208
- ":MARKer:X2Y2source" on page 209
- ":MARKer:XDELta" on page 210
- ":MARKer:Y1Position" on page 211
- ":MARKer:Y2Position" on page 212
- ":MARKer:YDELta" on page 213
- ":MEASure:CLEar" on page 220
- ":MEASure:COUNter" on page 221
- ":MEASure:DEFine" on page 222
- ":MEASure:DELay" on page 225
- ":MEASure:DUTYcycle" on page 227
- ":MEASure:FALLtime" on page 228
- ":MEASure:FREQuency" on page 229
- ":MEASure:LOWer" on page 478
- ":MEASure:NWIDth" on page 230
- ":MEASure:OVERshoot" on page 231
- ":MEASure:PERiod" on page 233
- ":MEASure:PHASe" on page 234
- ":MEASure:PREShoot" on page 235
- ":MEASure:PWIDth" on page 236
- ":MEASure:RISetime" on page 237
- ":MEASure:SCRatch" on page 479
- ":MEASure:SDEViation" on page 238
- ":MEASure:SHOW" on page 239
- ":MEASure:SOURce" on page 240
- ":MEASure:TDELta" on page 480
- ":MEASure:TEDGe" on page 242
- ":MEASure:THResholds" on page 481
- ":MEASure:TMAX" on page 482
- ":MEASure:TMIN" on page 483
- ":MEASure:TSTArt" on page 484
- ":MEASure:TSTOp" on page 485

- ":MEASure:TVALue" on page 244
- ":MEASure:TVOLt" on page 486
- ":MEASure:UPPer" on page 488
- ":MEASure:VAMPlitude" on page 246
- ":MEASure:VAVerage" on page 247
- ":MEASure:VBASe" on page 248
- ":MEASure:VDELta" on page 489
- ":MEASure:VMAX" on page 249
- ":MEASure:VMIN" on page 250
- ":MEASure:VPP" on page 251
- ":MEASure:VRMS" on page 252
- ":MEASure:VSTArt" on page 490
- ":MEASure:VSTOp" on page 491
- ":MEASure:VTIMe" on page 253
- ":MEASure:VTOP" on page 254
- ":MEASure:XMAX" on page 255
- ":MEASure:XMIN" on page 256
- ":MERGe" on page 104
- MODE Commands:
  - ":ACQuire:MODE" on page 129
  - ":MARKer:MODE" on page 205
  - ":SBUS:MODE" on page 286
  - ":TIMebase:MODE" on page 304
  - ":TRIGger:CAN:PATTern:ID:MODE" on page 329
  - ":TRIGger:MODE" on page 319
  - ":TRIGger:TV:MODE" on page 383
  - ":WAVeform:POINts:MODE" on page 415

N

- NREJect, ":TRIGger:NREJect" on page 320
- NWIDth, ":MEASure:NWIDth" on page 230

O

- OFFSet Commands:
  - ":CHANnel<n>:OFFSet" on page 151
  - ":FUNCtion:OFFSet" on page 185
- "*OPC (Operation Complete)" on page 72
- ":OPEE (Operation Status Enable Register)" on page 105

- OPERation, ":FUNCtion:OPERation" on page 186
- ":OPERegister:CONDition (Operation Status Condition Register)" on page 107
- ":OPERegister[:EVENt] (Operation Status Event Register)" on page 109
- "*OPT (Option Identification)" on page 73
- OVERload, ":SBUS:CAN:COUNt:OVERload" on page 279
- OVERshoot, ":MEASure:OVERshoot" on page 231
- ":OVLenable (Overload Event Enable Register)" on page 111
- ":OVLRegister (Overload Event Register)" on page 113

P

- PALette Commands:
  - ":HARDcopy:PALette" on page 200
  - ":SAVE:IMAGe:PALette" on page 270
- PARity Commands:
  - ":SBUS:LIN:PARity" on page 285
  - ":TRIGger:UART:PARity" on page 394
- PATTern Commands:
  - ":TRIGger:CAN:PATTern:DATA" on page 326
  - ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
  - ":TRIGger:CAN:PATTern:ID" on page 328
  - ":TRIGger:CAN:PATTern:ID:MODE" on page 329
  - ":TRIGger:DURation:PATTern" on page 338
  - ":TRIGger:IIC:PATTern:ADDRess" on page 356
  - ":TRIGger:IIC:PATTern:DATA" on page 357
  - ":TRIGger:IIC:PATTern:DATa2" on page 358
  - ":TRIGger:PATTern" on page 321
  - ":TRIGger:SPI:PATTern:DATA" on page 376
  - ":TRIGger:SPI:PATTern:WIDTh" on page 377
- PDRiver, ":HARDcopy:PDRiver" on page 477
- PERiod, ":MEASure:PERiod" on page 233
- PERSistence, ":DISPlay:PERSistence" on page 168
- PHASe, ":MEASure:PHASe" on page 234
- PMODe, ":CHANnel<n>:PMODe" on page 465
- POINts Commands:
  - ":ACQuire:POINts" on page 130
  - ":WAVeform:POINts" on page 413

  - ":WAVeform:POINts:MODE" on page 415
- POLarity Commands:
  - ":TRIGger:GLITch:POLarity" on page 351
  - ":TRIGger:TV:POLarity" on page 384
  - ":TRIGger:UART:POLarity" on page 395
- POSition Commands:
  - ":TIMebase:POSition" on page 305
  - ":TIMebase:WINDow:POSition" on page 310
- PREamble, ":WAVeform:PREamble" on page 417
- PREShoot, ":MEASure:PREShoot" on page 235
- ":PRINt" on page 115
- ":PRINt?" on page 492
- PRINter, ":HARDcopy:PRinter:LIST" on page 201
- PROBe Commands:
  - ":CHANnel<n>:PROBe" on page 152
  - ":EXTernal:PROBe" on page 175
- PROTection Commands:
  - ":CHANnel<n>:PROTection" on page 156
  - ":EXTernal:PROTection" on page 178
- Pulse Width (GLITch), ":TRIGger:GLITch Commands" on page 347
- PWD Commands:
  - ":RECall:PWD" on page 260
  - ":SAVE:PWD" on page 271
- PWIDth, ":MEASure:PWIDth" on page 236

Q

- QUALifier Commands:
  - ":TRIGger:DURation:QUALifier" on page 339
  - ":TRIGger:GLITch:QUALifier" on page 352
  - ":TRIGger:IIC:TRIGger:QUALifier" on page 361
  - ":TRIGger:UART:QUALifier" on page 396

R

- RANGe Commands:
  - ":CHANnel<n>:RANGe" on page 157
  - ":EXTernal:RANGe" on page 179
  - ":FUNCtion:RANGe" on page 187
  - ":TIMebase:RANGe" on page 306

  - ":TIMebase:WINDow:RANGe" on page 311
  - ":TRIGger:DURation:RANGe" on page 340
  - ":TRIGger:GLITch:RANGe" on page 353
- "*RCL (Recall)" on page 74
- ":RECall:FILename" on page 258
- ":RECall:IMAGe[:STARt]" on page 259
- ":RECall:PWD" on page 260
- ":RECall:SETup[:STARt]" on page 261
- REFerence Commands:
  - ":FUNCtion:REFerence" on page 188
  - ":TIMebase:REFerence" on page 307
- REJect, ":TRIGger[:EDGE]:REJect" on page 344
- RESet Commands:
  - ":SBUS:CAN:COUNt:RESet" on page 280
  - ":SBUS:UART:COUNt:RESet" on page 290
- RISetime, ":MEASure:RISetime" on page 237
- "Root (:) Commands" on page 86
- "*RST (Reset)" on page 75
- ":RUN" on page 116
- RX, ":TRIGger:UART:SOURce:RX" on page 397
- RXFRames, ":SBUS:UART:COUNt:RXFRames" on page 291

S

- SAMPlepoint Commands:
  - ":TRIGger:CAN:SAMPlepoint" on page 330
  - ":TRIGger:LIN:SAMPlepoint" on page 366
- "*SAV (Save)" on page 78
- ":SAVE:FILename" on page 264
- ":SAVE:IMAGe:AREA" on page 266
- ":SAVE:IMAGe:FACTors" on page 267
- ":SAVE:IMAGe:FORMat" on page 268
- ":SAVE:IMAGe:INKSaver" on page 269
- ":SAVE:IMAGe:PALette" on page 270
- ":SAVE:IMAGe[:STARt]" on page 265
- ":SAVE:PWD" on page 271
- ":SAVE:SETup[:STARt]" on page 272
- ":SAVE:WAVeform:FORMat" on page 274

- ":SAVE:WAVeform:LENGth" on page 275
- ":SAVE:WAVeform[:STARt]" on page 273
- ":SBUS:CAN:COUNt:ERRor" on page 278
- ":SBUS:CAN:COUNt:OVERload" on page 279
- ":SBUS:CAN:COUNt:RESet" on page 280
- ":SBUS:CAN:COUNt:TOTal" on page 281
- ":SBUS:CAN:COUNt:UTILization" on page 282
- ":SBUS:DISPlay" on page 283
- ":SBUS:IIC:ASIZe" on page 284
- ":SBUS:LIN:PARity" on page 285
- ":SBUS:MODE" on page 286
- ":SBUS:SPI:WIDTh" on page 287
- ":SBUS:UART:BASE" on page 288
- ":SBUS:UART:COUNt:ERRor" on page 289
- ":SBUS:UART:COUNt:RESet" on page 290
- ":SBUS:UART:COUNt:RXFRames" on page 291
- ":SBUS:UART:COUNt:TXFRames" on page 292
- ":SBUS:UART:FRAMing" on page 293
- SCALe Commands:
  - ":CHANnel<n>:SCALe" on page 158
  - ":FUNCtion:SCALe" on page 189
  - ":TIMebase:SCALe" on page 308
  - ":TIMebase:WINDow:SCALe" on page 312
- SCRatch, ":MEASure:SCRatch" on page 479
- SDEViation, ":MEASure:SDEViation" on page 238
- ":SERial" on page 117
- SETup Commands:
  - ":RECall:SETup[:STARt]" on page 261
  - ":SAVE:SETup[:STARt]" on page 272
  - ":SYSTem:SETup" on page 299
- SHOW, ":MEASure:SHOW" on page 239
- SIGNal Commands:
  - ":TRIGger:CAN:SIGNal:BAUDrate" on page 331
  - ":TRIGger:CAN:SIGNal:DEFinition" on page 496
  - ":TRIGger:LIN:SIGNal:BAUDrate" on page 367

  - ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":SINGle" on page 118
- SKEW, ":CHANnel<n>:PROBe:SKEW" on page 154
- SLOPe Commands:
  - ":TRIGger[:EDGE]:SLOPe" on page 345
  - ":TRIGger:SPI:CLOCk:SLOPe" on page 373
- SOURce Commands:
  - ":DISPlay:SOURce" on page 169
  - ":FUNCtion:SOURce" on page 190
  - ":MEASure:SOURce" on page 240
  - ":TRIGger:CAN:SOURce" on page 332
  - ":TRIGger:GLITch:SOURce" on page 354
  - ":TRIGger:IIC:SOURce:CLOCk" on page 359
  - ":TRIGger:IIC:SOURce:DATA" on page 360
  - ":TRIGger:LIN:SOURce" on page 368
  - ":TRIGger:SPI:SOURce:CLOCk" on page 378
  - ":TRIGger:SPI:SOURce:DATA" on page 379
  - ":TRIGger:SPI:SOURce:FRAMe" on page 380
  - ":TRIGger:TV:SOURce" on page 385
  - ":TRIGger:UART:SOURce:RX" on page 397
  - ":TRIGger:UART:SOURce:TX" on page 398
  - ":WAVeform:SOURce" on page 420
  - ":WAVeform:SOURce:SUBSource" on page 424
- SPAN, ":FUNCtion:SPAN" on page 191
- SPI Commands:
  - ":SBUS:SPI:WIDTh" on page 287
  - ":TRIGger:SPI Commands" on page 372
- SRATe, ":ACQuire:SRATe" on page 131
- "*SRE (Service Request Enable)" on page 79
- STANdard Commands:
  - ":TRIGger:LIN:STANdard" on page 369
  - ":TRIGger:TV:STANdard" on page 386
- STARt Commands:
  - ":CALibrate:STARt" on page 137
  - ":HARDcopy:STARt" on page 202

    - ":RECall:IMAGe[:STARt]" on page 259
    - ":RECall:SETup[:STARt]" on page 261
    - ":SAVE:IMAGe[:STARt]" on page 265
    - ":SAVE:SETup[:STARt]" on page 272
    - ":SAVE:WAVeform[:STARt]" on page 273
- STATus Commands:
    - ":CALibrate:STATus" on page 138
    - ":STATus" on page 119
- "*STB (Read Status Byte)" on page 81
- ":STOP" on page 120
- SUBSource, ":WAVeform:SOURce:SUBSource" on page 424
- SWEep, ":TRIGger:SWEep" on page 323
- SWITch, ":CALibrate:SWITch" on page 139
- SYNCbreak, ":TRIGger:LIN:SYNCbreak" on page 370
- ":SYSTem:DATE" on page 295
- ":SYSTem:DSP" on page 296
- ":SYSTem:ERRor" on page 297
- ":SYSTem:LOCK" on page 298
- ":SYSTem:SETup" on page 299
- ":SYSTem:TIME" on page 301

T

- TDELta, ":MEASure:TDELta" on page 480
- TEDGe, ":MEASure:TEDGe" on page 242
- TEMPerature, ":CALibrate:TEMPerature" on page 140
- ":TER (Trigger Event Register)" on page 121
- THResholds, ":MEASure:THResholds" on page 481
- TIME Commands:
    - ":CALibrate:TIME" on page 141
    - ":SYSTem:TIME" on page 301
- ":TIMebase:DELay" on page 494
- ":TIMebase:MODE" on page 304
- ":TIMebase:POSition" on page 305
- ":TIMebase:RANGe" on page 306
- ":TIMebase:REFerence" on page 307
- ":TIMebase:SCALe" on page 308
- ":TIMebase:VERNier" on page 309

- ":TIMebase:WINDow:POSition" on page 310
- ":TIMebase:WINDow:RANGe" on page 311
- ":TIMebase:WINDow:SCALe" on page 312
- TIMeout, ":TRIGger:SPI:CLOCk:TIMeout" on page 374
- TMAX, ":MEASure:TMAX" on page 482
- TMIN, ":MEASure:TMIN" on page 483
- TOTal, ":SBUS:CAN:COUNt:TOTal" on page 281
- "*TRG (Trigger)" on page 83
- TRIGger Commands:
  - ":TRIGger:CAN:TRIGger" on page 333
  - ":TRIGger:IIC:TRIGger:QUALifier" on page 361
  - ":TRIGger:IIC:TRIGger[:TYPE]" on page 362
  - ":TRIGger:LIN:TRIGger" on page 371
- ":TRIGger:HFReject" on page 317
- ":TRIGger:HOLDoff" on page 318
- ":TRIGger:MODE" on page 319
- ":TRIGger:NREJect" on page 320
- ":TRIGger:PATTern" on page 321
- ":TRIGger:SWEep" on page 323
- ":TRIGger:CAN:ACKNowledge" on page 495
- ":TRIGger:CAN:PATTern:DATA" on page 326
- ":TRIGger:CAN:PATTern:DATA:LENGth" on page 327
- ":TRIGger:CAN:PATTern:ID" on page 328
- ":TRIGger:CAN:PATTern:ID:MODE" on page 329
- ":TRIGger:CAN:SAMPlepoint" on page 330
- ":TRIGger:CAN:SIGNal:BAUDrate" on page 331
- ":TRIGger:CAN:SIGNal:DEFinition" on page 496
- ":TRIGger:CAN:SOURce" on page 332
- ":TRIGger:CAN:TRIGger" on page 333
- ":TRIGger:DURation:GREaterthan" on page 336
- ":TRIGger:DURation:LESSthan" on page 337
- ":TRIGger:DURation:PATTern" on page 338
- ":TRIGger:DURation:QUALifier" on page 339
- ":TRIGger:DURation:RANGe" on page 340
- ":TRIGger[:EDGE]:COUPling" on page 342

- ":TRIGger[:EDGE]:LEVel" on page 343
- ":TRIGger[:EDGE]:REJect" on page 344
- ":TRIGger[:EDGE]:SLOPe" on page 345
- ":TRIGger[:EDGE]:SOURce" on page 346
- ":TRIGger:GLITch:GREaterthan" on page 348
- ":TRIGger:GLITch:LESSthan" on page 349
- ":TRIGger:GLITch:LEVel" on page 350
- ":TRIGger:GLITch:POLarity" on page 351
- ":TRIGger:GLITch:QUALifier" on page 352
- ":TRIGger:GLITch:RANGe" on page 353
- ":TRIGger:GLITch:SOURce" on page 354
- ":TRIGger:HFReject" on page 317
- ":TRIGger:HOLDoff" on page 318
- ":TRIGger:IIC:PATTern:ADDRess" on page 356
- ":TRIGger:IIC:PATTern:DATA" on page 357
- ":TRIGger:IIC:PATTern:DATa2" on page 358
- ":TRIGger:IIC:SOURce:CLOCk" on page 359
- ":TRIGger:IIC:SOURce:DATA" on page 360
- ":TRIGger:IIC:TRIGger:QUALifier" on page 361
- ":TRIGger:IIC:TRIGger[:TYPE]" on page 362
- ":TRIGger:LIN:SIGNal:BAUDrate" on page 367
- ":TRIGger:LIN:SIGNal:DEFinition" on page 497
- ":TRIGger:LIN:SOURce" on page 368
- ":TRIGger:LIN:TRIGger" on page 371
- ":TRIGger:MODE" on page 319
- ":TRIGger:NREJect" on page 320
- ":TRIGger:PATTern" on page 321
- ":TRIGger:SPI:CLOCk:SLOPe" on page 373
- ":TRIGger:SPI:CLOCk:TIMeout" on page 374
- ":TRIGger:SPI:FRAMing" on page 375
- ":TRIGger:SPI:PATTern:DATA" on page 376
- ":TRIGger:SPI:PATTern:WIDTh" on page 377
- ":TRIGger:SPI:SOURce:CLOCk" on page 378
- ":TRIGger:SPI:SOURce:DATA" on page 379
- ":TRIGger:SPI:SOURce:FRAMe" on page 380

- ":TRIGger:SWEep" on page 323
- ":TRIGger:TV:LINE" on page 382
- ":TRIGger:TV:MODE" on page 383
- ":TRIGger:TV:POLarity" on page 384
- ":TRIGger:TV:SOURce" on page 385
- ":TRIGger:TV:STANdard" on page 386
- ":TRIGger:TV:TVMode" on page 498
- ":TRIGger:UART:BAUDrate" on page 389
- ":TRIGger:UART:BITorder" on page 390
- ":TRIGger:UART:BURSt" on page 391
- ":TRIGger:UART:DATA" on page 392
- ":TRIGger:UART:IDLE" on page 393
- ":TRIGger:UART:PARity" on page 394
- ":TRIGger:UART:POLarity" on page 395
- ":TRIGger:UART:QUALifier" on page 396
- ":TRIGger:UART:SOURce:RX" on page 397
- ":TRIGger:UART:SOURce:TX" on page 398
- ":TRIGger:UART:TYPE" on page 399
- ":TRIGger:UART:WIDTh" on page 400
- "*TST (Self Test)" on page 84
- TSTArt, ":MEASure:TSTArt" on page 484
- TSTOp, ":MEASure:TSTOp" on page 485
- TV, ":TRIGger:TV Commands" on page 381
- TVALue, ":MEASure:TVALue" on page 244
- TVOLt, ":MEASure:TVOLt" on page 486
- TX, ":TRIGger:UART:SOURce:TX" on page 398
- TXFRames, ":SBUS:UART:COUNt:TXFRames" on page 292
- TYPE Commands:
  - ":ACQuire:TYPE" on page 132
  - ":WAVeform:TYPE" on page 425
  - ":TRIGger:IIC:TRIGger[:TYPE]" on page 362
  - ":TRIGger:UART:TYPE" on page 399

U

- UART Commands:
  - ":SBUS:UART:BASE" on page 288
  - ":SBUS:UART:COUNt:ERRor" on page 289

    - ":SBUS:UART:COUNt:RESet" on page 290
    - ":SBUS:UART:COUNt:RXFRames" on page 291
    - ":SBUS:UART:COUNt:TXFRames" on page 292
    - ":SBUS:UART:FRAMing" on page 293
    - ":TRIGger:UART:BAUDrate" on page 389
    - ":TRIGger:UART:BITorder" on page 390
    - ":TRIGger:UART:BURSt" on page 391
    - ":TRIGger:UART:DATA" on page 392
    - ":TRIGger:UART:IDLE" on page 393
    - ":TRIGger:UART:PARity" on page 394
    - ":TRIGger:UART:POLarity" on page 395
    - ":TRIGger:UART:QUALifier" on page 396
    - ":TRIGger:UART:SOURce:RX" on page 397
    - ":TRIGger:UART:SOURce:TX" on page 398
    - ":TRIGger:UART:TYPE" on page 399
    - ":TRIGger:UART:WIDTh" on page 400
- UNITs Commands:
    - ":CHANnel<n>:UNITs" on page 159
    - ":EXTernal:UNITs" on page 180
- UNSigned, ":WAVeform:UNSigned" on page 426
- UPPer, ":MEASure:UPPer" on page 488
- UTILization, ":SBUS:CAN:COUNt:UTILization" on page 282

V

- VAMPlitude, ":MEASure:VAMPlitude" on page 246
- VAVerage, ":MEASure:VAVerage" on page 247
- VBASe, ":MEASure:VBASe" on page 248
- VDELta, ":MEASure:VDELta" on page 489
- VECTors, ":DISPlay:VECTors" on page 170
- VERNier, ":CHANnel<n>:VERNier" on page 160
- ":VIEW" on page 122
- VMAX, ":MEASure:VMAX" on page 249
- VMIN, ":MEASure:VMIN" on page 250
- VPP, ":MEASure:VPP" on page 251
- VRMS, ":MEASure:VRMS" on page 252
- VSTArt, ":MEASure:VSTArt" on page 490
- VSTOp, ":MEASure:VSTOp" on page 491

- VTIMe, ":MEASure:VTIMe" on page 253
- VTOP, ":MEASure:VTOP" on page 254

W

- "*WAI (Wait To Continue)" on page 85
- WAVeform Commands:
  - ":SAVE:WAVeform:FORMat" on page 274
  - ":SAVE:WAVeform:LENGth" on page 275
  - ":SAVE:WAVeform[:STARt]" on page 273
- ":WAVeform:BYTeorder" on page 408
- ":WAVeform:COUNt" on page 409
- ":WAVeform:DATA" on page 410
- ":WAVeform:FORMat" on page 412
- ":WAVeform:POINts" on page 413
- ":WAVeform:POINts:MODE" on page 415
- ":WAVeform:PREamble" on page 417
- ":WAVeform:SOURce" on page 420
- ":WAVeform:SOURce:SUBSource" on page 424
- ":WAVeform:TYPE" on page 425
- ":WAVeform:UNSigned" on page 426
- ":WAVeform:VIEW" on page 427
- ":WAVeform:XINCrement" on page 428
- ":WAVeform:XORigin" on page 429
- ":WAVeform:XREFerence" on page 430
- ":WAVeform:YINCrement" on page 431
- ":WAVeform:YORigin" on page 432
- ":WAVeform:YREFerence" on page 433
- WIDTh Commands:
  - ":SBUS:SPI:WIDTh" on page 287
  - ":TRIGger:SPI:PATTern:WIDTh" on page 377
  - ":TRIGger:UART:WIDTh" on page 400
- WINDow, ":FUNCtion:WINDow" on page 192

X

- X1Position, ":MARKer:X1Position" on page 206
- X1Y1source, ":MARKer:X1Y1source" on page 207
- X2Position, ":MARKer:X2Position" on page 208
- X2Y2source, ":MARKer:X2Y2source" on page 209

- XDELta, ":MARKer:XDELta" on page 210
- XINCrement, ":WAVeform:XINCrement" on page 428
- XMAX, ":MEASure:XMAX" on page 255
- XMIN, ":MEASure:XMIN" on page 256
- XORigin, ":WAVeform:XORigin" on page 429
- XREFerence, ":WAVeform:XREFerence" on page 430

Y

- Y1Position, ":MARKer:Y1Position" on page 211
- Y2Position, ":MARKer:Y2Position" on page 212
- YDELta, ":MARKer:YDELta" on page 213
- YINCrement, ":WAVeform:YINCrement" on page 431
- YORigin, ":WAVeform:YORigin" on page 432
- YREFerence, ":WAVeform:YREFerence" on page 433

5
Obsolete and Discontinued Commands

Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs (see"Obsolete Commands" on page 530).

Obsolete Command	Current Command Equivalent	Behavior Differences
ANALog<n>:BWLimit	:CHANnel<n>:BWLimit (see page 145)	
ANALog<n>:COUPling	:CHANnel<n>:COUPling (see page 146)	
ANALog<n>:INVert	:CHANnel<n>:INVert (see page 149)	
ANALog<n>:LABel	:CHANnel<n>:LABel (see page 150)	
ANALog<n>:OFFSet	:CHANnel<n>:OFFSet (see page 151)	
ANALog<n>:PROBe	:CHANnel<n>:PROBe (see page 152)	
ANALog<n>:PMODe	none	
ANALog<n>:RANGe	:CHANnel<n>:RANGe (see page 157)	
:CHANnel:LABel (see page 462)	:CHANnel<n>:LABel (see page 150)	
:CHANnel2:SKEW (see page 463)	:CHANnel<n>:PROBe:SKEW (see page 154)	
:CHANnel<n>:INPut (see page 464)	:CHANnel<n>:IMPedance (see page 148)	
:CHANnel<n>:PMODe (see page 465)	none	
:DISPlay:CONNect (see page 466)	:DISPlay:VECTors (see page 170)	
:ERASe (see page 467)	:CDISplay (see page 95)	

Obsolete Command	Current Command Equivalent	Behavior Differences
:EXTernal:INPut (see page 468)	:EXTernal:IMPedance (see page 174)	
:EXTernal:PMODe (see page 469)	none	
FUNCtion1, FUNCtion2	:FUNCtion Commands (see page 181)	ADD not included
:FUNCtion:VIEW (see page 470)	:FUNCtion:DISPlay (see page 184)	
:HARDcopy:DESTination (see page 471)	:HARDcopy:FILename (see page 473)	
:HARDcopy:DEVice (see page 472)	:HARDcopy:FORMat (see page 474)	PLOTter, THINkjet not supported; TIF, BMP, CSV, SEIko added
:HARDcopy:FILename (see page 473)	:RECall:FILename (see page 258) :SAVE:FILename (see page 258)	
:HARDcopy:FORMat (see page 474)	:HARDcopy:APRinter (see page 196) :SAVE:IMAGe:FORMat (see page 268) :SAVE:WAVeform:FORMat (see page 274)	
:HARDcopy:GRAYscale (see page 475)	:HARDcopy:PALette (see page 200)	
:HARDcopy:IGColors (see page 476)	:HARDcopy:INKSaver (see page 199)	
:HARDcopy:PDRiver (see page 477)	:HARDcopy:APRinter (see page 196)	
:MEASure:LOWer (see page 478)	:MEASure:DEFine:THResholds (see page 222)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:SCRatch (see page 479)	:MEASure:CLEar (see page 220)	
:MEASure:TDELta (see page 480)	:MARKer:XDELta (see page 210)	
:MEASure:THResholds (see page 481)	:MEASure:DEFine:THResholds (see page 222)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:TMAX (see page 482)	:MEASure:XMAX (see page 255)	

Obsolete Command	Current Command Equivalent	Behavior Differences
:MEASure:TMIN (see page 483)	:MEASure:XMIN (see page 256)	
:MEASure:TSTArt (see page 484)	:MARKer:X1Position (see page 206)	
:MEASure:TSTOp (see page 485)	:MARKer:X2Position (see page 208)	
:MEASure:TVOLt (see page 486)	:MEASure:TVALue (see page 244)	TVALue measures additional values such as db, Vs, etc.
:MEASure:UPPer (see page 488)	:MEASure:DEFine:THResholds (see page 222)	MEASure:DEFine:THResholds can define absolute values or percentage
:MEASure:VDELta (see page 489)	:MARKer:YDELta (see page 213)	
:MEASure:VSTArt (see page 490)	:MARKer:Y1Position (see page 211)	
:MEASure:VSTOp (see page 491)	:MARKer:Y2Position (see page 212)	
:PRINt? (see page 492)	:DISPlay:DATA? (see page 164)	
:TIMebase:DELay (see page 494)	:TIMebase:POSition (see page 305) or :TIMebase:WINDow:POSition (see page 310)	TIMebase:POSition is position value of main time base; TIMebase:WINDow:POSition is position value of delayed time base window.
:TRIGger:CAN:ACKNowledge (see page 495)	none	
:TRIGger:CAN:SIGNal:DEFinition (see page 496)	none	
:TRIGger:LIN:SIGNal:DEFinition (see page 497)	none	
:TRIGger:TV:TVMode (see page 498)	:TRIGger:TV:MODE (see page 383)	

Discontinued Commands

Discontinued commands are commands that were used by previous oscilloscopes, but are not supported by the InfiniiVision 5000 Series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

Discontinued Command	Current Command Equivalent	Comments
ASTore	:DISPlay:PERSistence INFinite (see page 168)	
CHANnel:MATH	:FUNCtion:OPERation (see page 186)	ADD not included
CHANnel<n>:PROTect	:CHANnel<n>:PROTection (see page 156)	Previous form of this command was used to enable/disable 50Ω protection. The new command resets a tripped protect and the query returns the status of TRIPed or NORMal.
DISPlay:INVerse	none	
DISPlay:COLumn	none	
DISPlay:GRID	none	
DISPLay:LINE	none	
DISPlay:PIXel	none	
DISPlay:POSition	none	
DISPlay:ROW	none	
DISPlay:TEXT	none	
FUNCtion:MOVE	none	
FUNCtion:PEAKs	none	
HARDcopy:ADDRess	none	Only parallel printer port is supported. GPIB printing not supported
MASK	none	All commands discontinued, feature not available
SYSTem:KEY	none	
TEST:ALL	*TST (Self Test) (see page 84)	
TRACE subsystem	none	All commands discontinued, feature not available
TRIGger:ADVanced subsystem		Use new GLITch, PATTern, or TV trigger modes
TRIGger:TV:FIELd	:TRIGger:TV:MODE (see page 383)	
TRIGger:TV:TVHFrej		
TRIGger:TV:VIR	none	
VAUToscale	none	

Discontinued Parameters Some previous oscilloscope queries returned control setting values of OFF and ON. The InfiniiVision 5000 Series oscilloscopes only return the enumerated values 0 (for off) and 1 (for on).

:CHANnel:LABel

O (see page 530)

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 150) instead.

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
```

:CHANnel2:SKEW

O (see page 530)

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 154) 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 143

:CHANNel<n>:INPut

O (see page 530)

Command Syntax

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

NOTE The :CHANnel<n>:INPut command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :CHANnel<n>:IMPedance command (see page 148) instead.

Query Syntax

```
:CHANnel<n>:INPut?
```

The :CHANnel<n>:INPut? query returns the current input impedance setting for the specified channel.

Return Format

```
<impedance value><NL>

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

:CHANnel<n>:PMODe

O (see page 530)

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

O (see page 530)

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 170) 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 170

:ERASe

O (see page 530)

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 :CDISplay command (see page 95) instead.

:EXTernal:INPut

O (see page 530)

Command Syntax

```
:EXTernal:INPut <impedance>
```

```
<impedance> ::= {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 The :EXTernal:INPut command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :EXTernal:IMPedance command (see page 174) 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> ::= {ONEM | FIFT}
```

See Also

- "Introduction to :EXTernal Trigger Commands" on page 171
- "Introduction to :TRIGger Commands" on page 313
- ":CHANnel<n>:IMPedance" on page 148

:EXTernal:PMODe

O (see page 530)

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

O (see page 530)

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 184) 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

0 (see page 530)

Command Syntax

```
:HARDcopy:DESTination <destination>

<destination> ::= {CENTronics | FLOPpy}
```

The :HARDcopy:DESTination command sets the hardcopy destination.

NOTE The :HARDcopy:DESTination command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:FILename command (see page 473) instead.

Query Syntax

```
:HARDcopy:DESTination?
```

The :HARDcopy:DESTination? query returns the selected hardcopy destination.

Return Format

```
<destination><NL>

<destination> ::= {CENT | FLOP}
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:FORMat" on page 474

:HARDcopy:DEVice

O (see page 530)

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 474) 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}
```

:HARDcopy:FILename

O (see page 530)

Command Syntax

```
:HARDcopy:FILename <string>
```

```
<string> ::= quoted ASCII string
```

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

NOTE The :HARDcopy:FILename command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :SAVE:FILename command (see page 264) and :RECall:FILename command (see page 258) instead.

Query Syntax

```
:HARDcopy:FILename?
```

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

Return Format

```
<string><NL>
```

```
<string> ::= quoted ASCII string
```

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:FORMat" on page 474

:HARDcopy:FORMat

O (see page 530)

Command Syntax

```
:HARDcopy:FORMat <format>

<format> ::= {BMP[24bit] | BMP8bit | PNG | CSV | ASCiixy | BINary
             | PRINter0 | PRINter1}
```

The HARDcopy:FORMat command sets the hardcopy format type.

PRINter0 and PRINter1 are only valid when printers are connected to the oscilloscope's USB ports. (The first printer connected/identified is PRINter0 and the second is PRINter1.)

NOTE The :HARDcopy:FORMat command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :SAVE:IMAGe:FORMat (see page 268), :SAVE:WAVeform:FORMat (see page 274), and :HARDcopy:APRinter (see page 196) commands instead.

Query Syntax

```
:HARDcopy:FORMat?
```

The :HARDcopy:FORMat? query returns the selected hardcopy format type.

Return Format

```
<format><NL>

<format> ::= {BMP | BMP8 | PNG | CSV | ASC | BIN | PRIN0 | PRIN1}
```

See Also

- "Introduction to :HARDcopy Commands" on page 193

:HARDcopy:GRAYscale

O (see page 530)

Command Syntax

```
:HARDcopy:GRAYscale <gray>
```

```
<gray> ::= {{OFF | 0} | {ON | 1}}
```

The :HARDcopy:GRAYscale command controls whether grayscaling is performed in the hardcopy dump.

NOTE The :HARDcopy:GRAYscale command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:PALette command (see page 200) instead. (":HARDcopy:GRAYscale ON" is the same as ":HARDcopy:PALette GRAYscale" and ":HARDcopy:GRAYscale OFF" is the same as ":HARDcopy:PALette COLor".)

Query Syntax

```
:HARDcopy:GRAYscale?
```

The :HARDcopy:GRAYscale? query returns a flag indicating whether grayscaling is performed in the hardcopy dump.

Return Format

```
<gray><NL>
```

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

See Also

- "Introduction to :HARDcopy Commands" on page 193

:HARDcopy:IGColors

O (see page 530)

Command Syntax

```
:HARDcopy:IGColors <value>

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

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

NOTE

The :HARDcopy:IGColors command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:INKSaver (see page 199) command instead.

Query Syntax

```
:HARDcopy:IGColors?
```

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

Return Format

```
<value><NL>

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

See Also

- "Introduction to :HARDcopy Commands" on page 193

:HARDcopy:PDRiver

O (see page 530)

Command Syntax

```
:HARDcopy:PDRiver <driver>

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

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

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

NOTE The :HARDcopy:PDRiver command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :HARDcopy:APRinter (see page 196) command instead.

Query Syntax

```
:HARDcopy:PDRiver?
```

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

Return Format

```
<driver><NL>

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

See Also

- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:FORMat" on page 474

:MEASure:LOWer

O (see page 530)

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 222) 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 218
- ":MEASure:THResholds" on page 481
- ":MEASure:UPPer" on page 488

:MEASure:SCRatch

O (see page 530)

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 220) instead.

:MEASure:TDELta

O (see page 530)

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 210) instead.

Return Format

```
<value><NL>

<value> ::= time difference between start and stop markers in NR3 format
```

See Also

- "Introduction to :MARKer Commands" on page 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:X1Position" on page 206
- ":MARKer:X2Position" on page 208
- ":MARKer:XDELta" on page 210
- ":MEASure:TSTArt" on page 484
- ":MEASure:TSTOp" on page 485

:MEASure:THResholds

O (see page 530)

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 222) 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 218
- ":MEASure:LOWer" on page 478
- ":MEASure:UPPer" on page 488

:MEASure:TMAX

O (see page 530)

Command Syntax

```
:MEASure:TMAX [<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:TMAX command installs a screen measurement and starts an X-at-Max-Y measurement on the selected waveform. If the optional source is specified, the current source is modified.

NOTE The :MEASure:TMAX command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :MEASure:XMAX command (see page 255) instead.

Query Syntax

```
:MEASure:TMAX? [<source>]
```

The :MEASure:TMAX? query returns the horizontal axis value at which the maximum vertical value occurs on the current source. If the optional source is specified, the current source is modified. If all channels are off, the query returns 9.9E+37.

Return Format

```
<value><NL>

<value> ::= time at maximum in NR3 format
```

See Also

- "Introduction to :MEASure Commands" on page 218
- ":MEASure:TMIN" on page 483
- ":MEASure:XMAX" on page 255
- ":MEASure:XMIN" on page 256

:MEASure:TMIN

O (see page 530)

Command Syntax

```
:MEASure:TMIN [<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: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 256) 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 218
- ":MEASure:TMAX" on page 482
- ":MEASure:XMAX" on page 255
- ":MEASure:XMIN" on page 256

:MEASure:TSTArt

O (see page 530)

Command Syntax

```
:MEASure:TSTArt <value> [suffix]
```

```
<value> ::= time at the start marker in seconds
```

```
[suffix] ::= {s | ms | us | ns | ps}
```

The :MEASure:TSTArt command moves the start marker (X1 cursor) to the specified time with respect to the trigger time.

NOTE The short form of this command, TSTA, does not follow the defined Long Form to Short Form Truncation Rules (see page 532). 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 206) instead.

Query Syntax

```
:MEASure:TSTArt?
```

The :MEASure:TSTArt? query returns the time at the start marker (X1 cursor).

Return Format

```
<value><NL>
```

```
<value> ::= time at the start marker in NR3 format
```

See Also

- "Introduction to :MARKer Commands" on page 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:X1Position" on page 206
- ":MARKer:X2Position" on page 208
- ":MARKer:XDELta" on page 210
- ":MEASure:TDELta" on page 480
- ":MEASure:TSTOp" on page 485

:MEASure:TSTOp

O (see page 530)

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 532). 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 208) 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 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:X1Position" on page 206
- ":MARKer:X2Position" on page 208
- ":MARKer:XDELta" on page 210
- ":MEASure:TDELta" on page 480
- ":MEASure:TSTArt" on page 484

:MEASure:TVOLt

0 (see page 530)

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> ::= {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: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 244) instead.

Return Format

```
<value><NL>
```

```
<value> ::= time in seconds of the specified voltage crossing
            in NR3 format
```

:MEASure:UPPer

O (see page 530)

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 222) 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 218
- ":MEASure:LOWer" on page 478
- ":MEASure:THResholds" on page 481

:MEASure:VDELta

O (see page 530)

Query Syntax

```
:MEASure:VDELta?
```

The :MEASure:VDELta? query returns the voltage difference between vertical marker 1 (Y1 cursor) and vertical marker 2 (Y2 cursor). No measurement is made when the :MEASure:VDELta? query is received by the oscilloscope. The delta value that is returned is the current value. This is the same value as the front-panel cursors delta Y value.

VDELta = value at marker 2 - 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 213) instead.

Return Format

```
<value><NL>

<value> ::= delta V value in NR1 format
```

See Also

- "Introduction to :MARKer Commands" on page 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:Y1Position" on page 211
- ":MARKer:Y2Position" on page 212
- ":MARKer:YDELta" on page 213
- ":MEASure:TDELta" on page 480
- ":MEASure:TSTArt" on page 484

:MEASure:VSTArt

O (see page 530)

Command Syntax

```
:MEASure:VSTArt <vstart_argument>

<vstart_argument> ::= value for vertical marker 1
```

The :MEASure:VSTArt command moves the vertical marker (Y1 cursor) to the specified value corresponding to the selected source. The source can be selected by the MARKer:X1Y1source command.

NOTE The short form of this command, VSTA, does not follow the defined Long Form to Short Form Truncation Rules (see page 532). 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 211) instead.

Query Syntax

```
:MEASure:VSTArt?
```

The :MEASure:VSTArt? query returns the current value of the Y1 cursor.

Return Format

```
<value><NL>

<value> ::= voltage at voltage marker 1 in NR3 format
```

See Also

- "Introduction to :MARKer Commands" on page 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:Y1Position" on page 211
- ":MARKer:Y2Position" on page 212
- ":MARKer:YDELta" on page 213
- ":MARKer:X1Y1source" on page 207
- ":MEASure:SOURce" on page 240
- ":MEASure:TDELta" on page 480
- ":MEASure:TSTArt" on page 484

:MEASure:VSTOp

O (see page 530)

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 532). 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 212) 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 204
- "Introduction to :MEASure Commands" on page 218
- ":MARKer:Y1Position" on page 211
- ":MARKer:Y2Position" on page 212
- ":MARKer:YDELta" on page 213
- ":MARKer:X2Y2source" on page 209
- ":MEASure:SOURce" on page 240
- ":MEASure:TDELta" on page 480
- ":MEASure:TSTArt" on page 484

:PRINt?

O (see page 530)

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 164) instead.

Print Option	:PRINt command	:PRINt? query	Query Default
COLor	Sets palette=COLor		
GRAYscale	Sets palette=GRAYscale		palette=COLor
PRINter0,1	Causes the USB printer #0,1 to be selected as destination (if connected)	Not used	N/A
BMP8bit	Sets print format to 8-bit BMP	Selects 8-bit BMP formatting for query	N/A
BMP	Sets print format to BMP	Selects BMP formatting for query	N/A
FACTors	Selects outputting of additional settings information for :PRINT	Not used	N/A
NOFactors	Deselects outputting of additional settings information for :PRINT	Not used	N/A

Old Print Option:	Is Now:
HIRes	COLor
LORes	GRAYscale
PARallel	PRINter0

Old Print Option:	Is Now:
DISK	invalid
PCL	invalid

NOTE The PRINt? query is not a core command.

See Also

- "Introduction to Root (:) Commands" on page 88
- "Introduction to :HARDcopy Commands" on page 193
- ":HARDcopy:FORMat" on page 474
- ":HARDcopy:FACTors" on page 197
- ":HARDcopy:GRAYscale" on page 475
- ":DISPlay:DATA" on page 164

:TIMebase:DELay

O (see page 530)

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 307).

NOTE The :TIMebase:DELay command is obsolete and is provided for backward compatibility to previous oscilloscopes. Use the :TIMebase:POSition command (see page 305) 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 578

:TRIGger:CAN:ACKNowledge

O (see page 530)

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 InfiniiVision 5000 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 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:TRIGger" on page 333

:TRIGger:CAN:SIGNal:DEFinition

O (see page 530)

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 InfiniiVision 5000 Series oscilloscope software version 5.00 or greater, this command is available, but the only legal value is DIFF.

Query Syntax

```
:TRIGger:CAN:SIGNal:DEFinition?
```

The :TRIGger:CAN:SIGNal:DEFinition? query returns the current CAN signal type.

Return Format

```
<value><NL>

<value> ::= DIFF
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:CAN:SIGNal:BAUDrate" on page 331
- ":TRIGger:CAN:SOURce" on page 332
- ":TRIGger:CAN:TRIGger" on page 333

:TRIGger:LIN:SIGNal:DEFinition

O (see page 530)

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 InfiniiVision 5000 Series oscilloscope software version 5.00 or greater, this command is available, but the only legal value is LIN.

Query Syntax

```
:TRIGger:LIN:SIGNal:DEFinition?
```

The :TRIGger:LIN:SIGNal:DEFinition? query returns the current LIN signal type.

Return Format

```
<value><NL>

<value> ::= LIN
```

See Also

- "Introduction to :TRIGger Commands" on page 313
- ":TRIGger:MODE" on page 319
- ":TRIGger:LIN:SIGNal:BAUDrate" on page 367
- ":TRIGger:LIN:SOURce" on page 368

:TRIGger:TV:TVMode

O (see page 530)

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 LVERtical parameter is only available when :TRIGger:TV:STANdard is GENeric. The LALTernate parameter is not available when :TRIGger:TV:STANdard is GENeric (see page 386).

Old forms for <mode> are accepted:

<mode>	Old Forms Accepted
FIEld1	F1
FIEld2	F2
AFIeld	ALLFields, ALLFLDS
ALINes	ALLLines
LFIeld1	LINEF1, LINEFIELD1
LFIeld2	LINEF2, LINEFIELD2
LALTernate	LINEAlt
LVERtical	LINEVert

NOTE The :TRIGger:TV:TVMode command is an obsolete command provided for compatibility to previous oscilloscopes. Use the :TRIGger:TV:MODE command (see page 383) instead.

Query Syntax

```
:TRIGger:TV:TVMode?
```

The :TRIGger:TV:TVMode? query returns the TV trigger mode.

Return Format

```
<value><NL>

<value> ::= {FIE1 | FIE2 | AFI | ALIN | LINE | VERT | LFI1 | LFI2
            | 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

-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

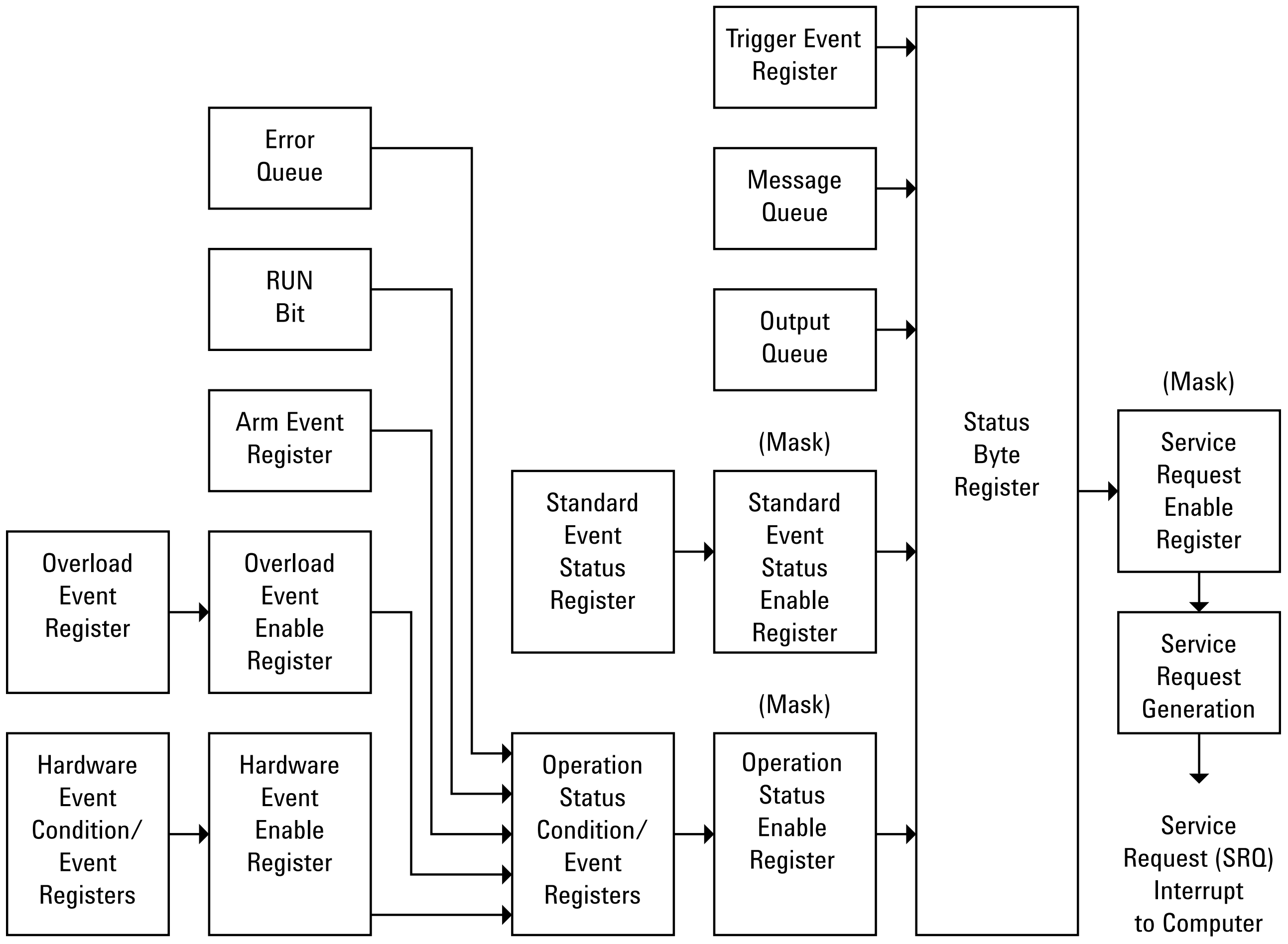
+113, Directory Not Supported

7
Status Reporting

Status Reporting Data Structures 509
Status Byte Register (STB) 512
Service Request Enable Register (SRE) 514
Trigger Event Register (TER) 515
Output Queue 516
Message Queue 517
(Standard) Event Status Register (ESR) 518
(Standard) Event Status Enable Register (ESE) 519
Error Queue 520
Operation Status Event Register (:OPERegister[:EVENt]) 521
Operation Status Condition Register (:OPERegister:CONDition) 522
Arm Event Register (AER) 523
Hardware Event Event Register (:HWERegister[:EVENt]) 524
Hardware Event Condition Register (:HWERegister:CONDition) 525
Clearing Registers and Queues 526
Status Reporting Decision Chart 527

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.


Trigger Event Register
Message Queue
Output Queue
Error Queue
RUN Bit
Arm Event Register
Status Byte Register
(Mask)
Service Request Enable Register
(Mask)
Standard Event Status Register
Standard Event Status Enable Register
Overload Event Register
Overload Event Enable Register
Service Request Generation
(Mask)
Hardware Event Condition/ Event Registers
Hardware Event Enable Register
Operation Status Condition/ Event Registers
Operation Status Enable Register
Service Request (SRQ) Interrupt to Computer

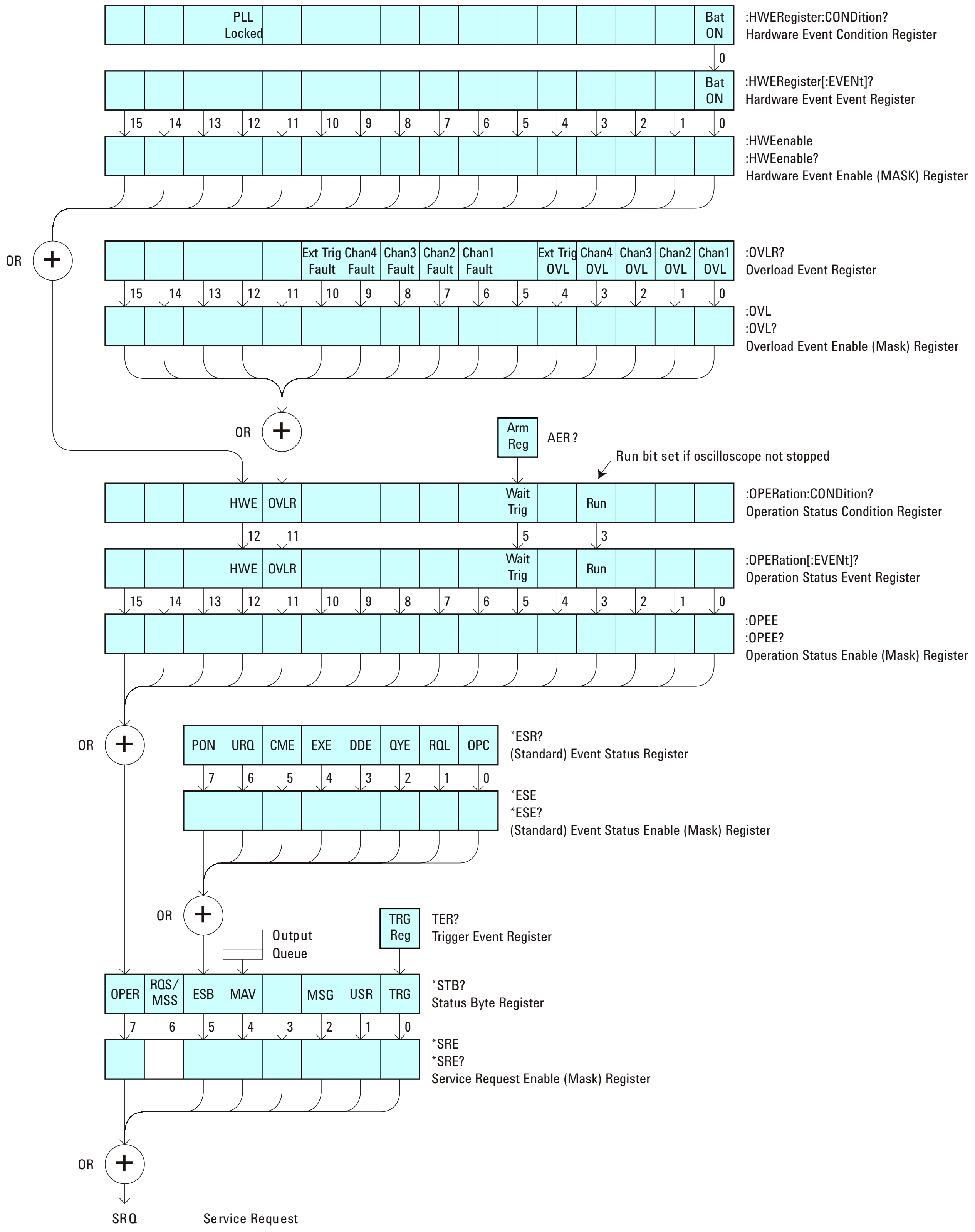
- To monitor an event, first clear the event; then, enable the event. All of the events are cleared when you initialize the instrument.
- To allow a service request (SRQ) interrupt to an external controller, enable at least one bit in the Status Byte Register (by setting, or unmasking, the bit in the 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.


PLL Locked
Bat ON
:HWERegister:CONDition?
Hardware Event Condition Register
:HWERegister[:EVENt]?
Hardware Event Event Register
:HWEenable
:HWEenable?
Hardware Event Enable (MASK) Register
OR
Ext Trig Fault
Chan4 Fault
Chan3 Fault
Chan2 Fault
Chan1 Fault
Ext Trig OVL
Chan4 OVL
Chan3 OVL
Chan2 OVL
Chan1 OVL
:OVLR?
Overload Event Register
:OVL
:OVL?
Overload Event Enable (Mask) Register
Arm Reg
AER?
Run bit set if oscilloscope not stopped
HWE
OVLR
Wait Trig
Run
:OPERation:CONDition?
Operation Status Condition Register
:OPERation[:EVENt]?
Operation Status Event Register
:OPEE
:OPEE?
Operation Status Enable (Mask) Register
PON
URQ
CME
EXE
DDE
QYE
RQL
OPC
*ESR?
(Standard) Event Status Register
*ESE
*ESE?
(Standard) Event Status Enable (Mask) Register
Output Queue
TRG Reg
TER?
Trigger Event Register
OPER
RQS/ MSS
ESB
MAV
MSG
USR
TRG
*STB?
Status Byte Register
*SRE
*SRE?
Service Request Enable (Mask) Register
SRQ
Service Request

The status register bits are described in more detail in the following tables:

- "Status Byte Register (STB)" on page 81
- "Standard Event Status Register (ESR)" on page 68
- "Operation Status Condition Register" on page 107
- "Operation Status Event Register" on page 109
- "Overload Event Register (OVLR)" on page 113
- "Hardware Event Condition Register" on page 100
- "Hardware Event Event Register" on page 102

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.

```
myScope.WriteString "*STB?"
varQueryResult = myScope.ReadNumber
MsgBox "Status Byte Register, Read: 0x" + Hex(varQueryResult)
```

The next program prints 0xD1 and clears bit 6 (RQS) and bit 4 (MAV) of the Status Byte Register. The difference in the output value between this example and the previous one is the value of bit 6 (weight = 64). Bit 6 is set when the first enabled summary bit is set and is cleared when the Status Byte Register is read by the serial poll command.

Example The following example uses the resource session object's ReadSTB method to read the contents of the oscilloscope's Status Byte Register.

```
varQueryResult = myScope.IO.ReadSTB
MsgBox "Status Byte Register, Serial Poll: 0x" + Hex(varQueryResult)
```

NOTE **Use Serial Polling to Read Status Byte Register**. Serial polling is the preferred method to read the contents of the Status Byte Register because it resets bit 6 and allows the next enabled event that occurs to generate a new SRQ interrupt.

Service Request Enable Register (SRE)

Setting the Service Request Enable Register bits enable corresponding bits in the Status Byte Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

Bits are set in the Service Request Enable Register using the *SRE command and the bits that are set are read with the *SRE? query.

Example The following example sets bit 4 (MAV) and bit 5 (ESB) in the Service Request Enable Register.

```
myScope.WriteString "*SRE " + CStr(CInt("&H30"))
```

This example uses the decimal parameter value of 48, the string returned by CStr(CInt("&H30")), to enable the oscilloscope to generate an SRQ interrupt under the following conditions:

- When one or more bytes in the Output Queue set bit 4 (MAV).
- When an enabled event in the Standard Event Status Register generates a summary bit that sets bit 5 (ESB).

Trigger Event Register (TER)

This register sets the TRG bit in the status byte when a trigger event occurs.

The TER event register stays set until it is cleared by reading the register or using the *CLS command. If your application needs to detect multiple triggers, the TER event register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation, you must clear the event register each time the trigger bit is set.

Output Queue

The output queue stores the oscilloscope-to-controller responses that are generated by certain instrument commands and queries. The output queue generates the Message Available summary bit when the output queue contains one or more bytes. This summary bit sets the MAV bit (bit 4) in the Status Byte Register.

When using the 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:DSP command do not set the MSG status bit in the Status Byte Register.

(Standard) Event Status Register (ESR)

The (Standard) Event Status Register (ESR) monitors the following oscilloscope status events:

- PON - Power On
- URQ - User Request
- CME - Command Error
- EXE - Execution Error
- DDE - Device Dependent Error
- QYE - Query Error
- RQC - Request Control
- OPC - Operation Complete

When one of these events occur, the event sets the corresponding bit in the register. If the bits are enabled in the Standard Event Status Enable Register, the bits set in this register generate a summary bit to set bit 5 (ESB) in the Status Byte Register.

You can read the contents of the Standard Event Status Register and clear the register by sending the *ESR? query. The value returned is the total bit weights of all of the bits that are set at the present time.

Example The following example uses the *ESR query to read the contents of the Standard Event Status Register.

```
myScope.WriteString "*ESR?"
varQueryResult = myScope.ReadNumber
MsgBox "Standard Event Status Register: 0x" + Hex(varQueryResult)
```

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights.

(Standard) Event Status Enable Register (ESE)

To allow any of the (Standard) Event Status Register (ESR) bits to generate a summary bit, you must first enable that bit. Enable the bit by using the *ESE (Event Status Enable) common command to set the corresponding bit in the (Standard) Event Status Enable Register (ESE).

Set bits are read with the *ESE? query.

Example Suppose your application requires an interrupt whenever any type of error occurs. The error related bits in the (Standard) Event Status Register are bits 2 through 5 (hexadecimal value 0x3C). Therefore, you can enable any of these bits to generate the summary bit by sending:

```
myScope.WriteString "*ESE " + CStr(CInt("&H3C"))
```

Whenever an error occurs, it sets one of these bits in the (Standard) Event Status Register. Because all the error related bits are enabled, a summary bit is generated to set bit 5 (ESB) in the Status Byte Register.

If bit 5 (ESB) in the Status Byte Register is enabled (via the *SRE command), an SRQ service request interrupt is sent to the controller PC.

NOTE

Disabled (Standard) Event Status Register bits respond but do not generate a summary bit. (Standard) Event Status Register bits that are not enabled still respond to their corresponding conditions (that is, they are set if the corresponding event occurs). However, because they are not enabled, they do not generate a summary bit to the Status Byte Register.

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error 350, Queue overflow. Any time the queue overflows, the least recent errors remain in the queue, and the most recent error is discarded. The length of the oscilloscope's error queue is 30 (29 positions for the error messages, and 1 position for the Queue overflow message).

The error queue is read with the :SYSTem:ERRor? query. Executing this query reads and removes the oldest error from the head of the queue, which opens a position at the tail of the queue for a new error. When all the errors have been read from the queue, subsequent error queries return "0, No error".

The error queue is cleared when:

- the instrument is powered up,
- the instrument receives the *CLS common command, or
- the last item is read from the error queue.

Operation Status Event Register (:OPERegister[:EVENt])

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 (:HWERegister[:EVENt])

This register hosts the PLL LOCKED bit (bit 12).

- The PLL LOCKED bit (bit 12) is for internal use and is not intended for general use.

Hardware Event Condition Register (:HWERegister:CONDition)

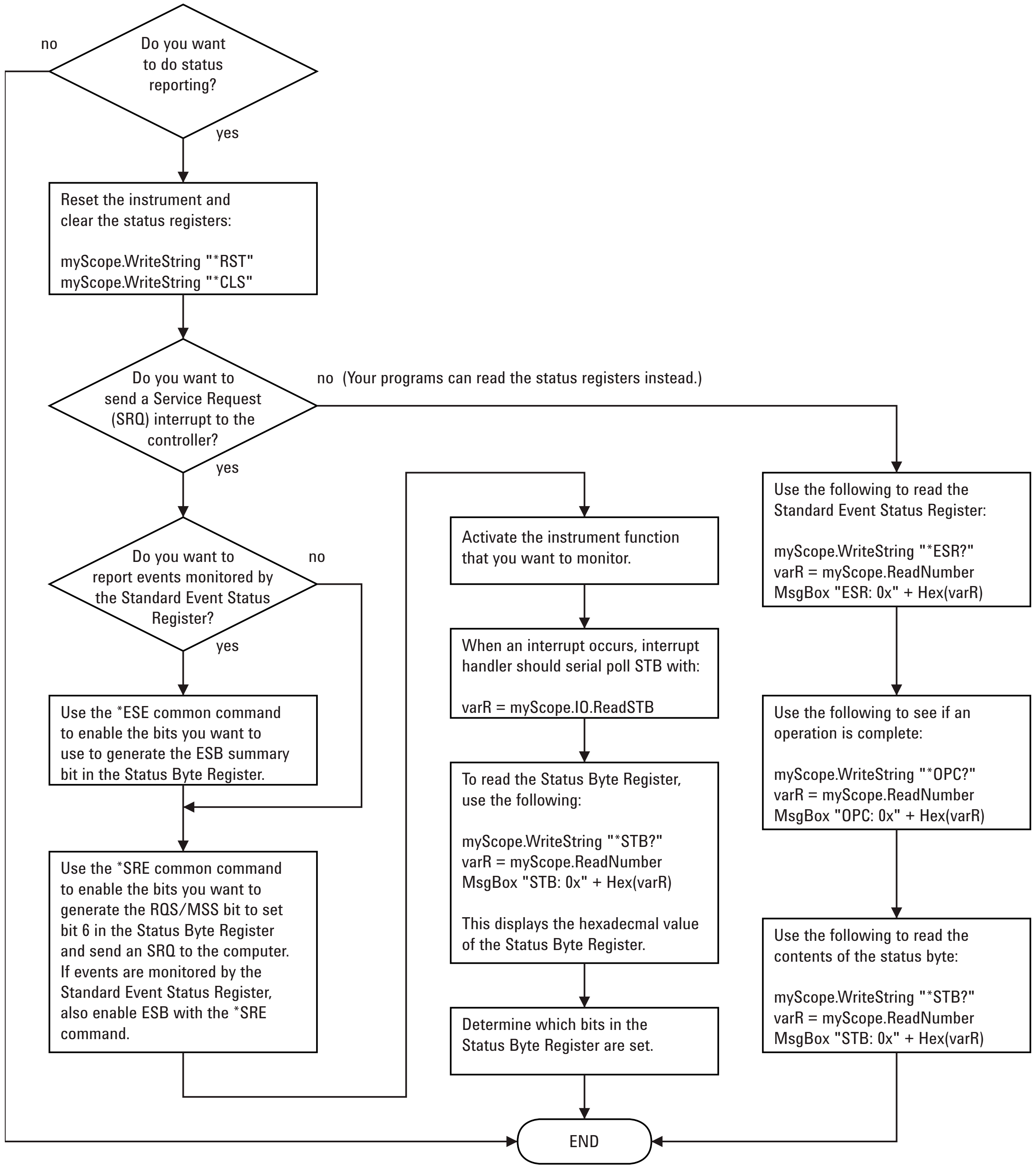
This register hosts the PLL LOCKED bit (bit 12).

- The :HWERegister:CONDition? query returns the value of the Hardware Event Condition Register.
- The PLL LOCKED bit (bit 12) is for internal use and is not intended for general use.

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


no
Do you want to do status reporting?
yes
Reset the instrument and clear the status registers:
myScope.WriteString "*RST"
myScope.WriteString "*CLS"
Do you want to send a Service Request (SRQ) interrupt to the controller?
no (Your programs can read the status registers instead.)
yes
Do you want to report events monitored by the Standard Event Status Register?
no
yes
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 RQS/MSS bit to set bit 6 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.
Activate the instrument function that you want to monitor.
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 hexadecmal value of the Status Byte Register.
Determine which bits in the Status Byte Register are set.
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

8
More About Oscilloscope Commands

Command Classifications 530
Valid Command/Query Strings 531
Query Return Values 547
All Oscilloscope Commands Are Sequential 548

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 530
- "Non-Core Commands" on page 530
- "Obsolete Commands" on page 530

C 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.

N 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.

O Obsolete Commands

Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs. Generally, these commands are mapped onto some of the Core and Non-core commands, but may not strictly have the same behavior as the new command. None of the obsolete commands are guaranteed to remain functional in future products. New systems and programs should use the Core (and Non-core) commands. Obsolete commands are listed in:

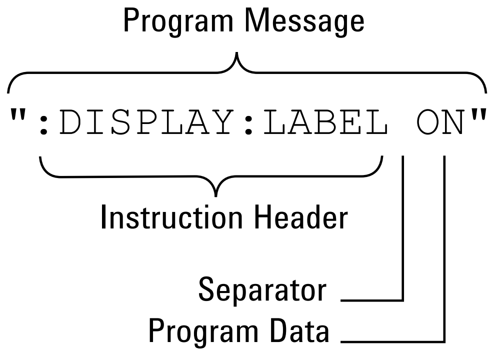
- "Obsolete and Discontinued Commands" on page 457
- As well as: "Commands A-Z" on page 435

Valid Command/Query Strings

- "Program Message Syntax" on page 531
- "Command Tree" on page 535
- "Duplicate Mnemonics" on page 545
- "Tree Traversal Rules and Multiple Commands" on page 545

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 532), and upper and/or lower case ASCII characters may be used. (Query responses, however, are always returned in upper case.)

Instructions are composed of two main parts:

- The header, which specifies the command or query to be sent.
- The program data, which provide additional information needed to clarify the meaning of the instruction.

Instruction Header The instruction header is one or more mnemonics separated by colons (:) that represent the operation to be performed by the instrument. The "Command Tree" on page 535 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 533
- "Compound Command Headers" on page 533
- "Common Command Headers" on page 534

White Space (Separator) White space is used to separate the instruction header from the program data. If the instruction does not require any program data parameters, you do not need to include any white space. White space is defined as one or more space characters. ASCII defines a space to be character 32 (in decimal).

Program Data Program data are used to clarify the meaning of the command or query. They provide necessary information, such as whether a function should be on or off, or which waveform is to be displayed. Each instruction's syntax definition shows the program data, as well as the values they accept. "Program Data Syntax Rules" on page 534 describes all of the general rules about acceptable values.

When there is more than one data parameter, they are separated by commas(,). Spaces can be added around the commas to improve readability.

Program Message Terminator The program instructions within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Or-Identify) asserted in the 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.

Long Form	Short form
RANGe	RANG
PATTern	PATT
TIMebase	TIM
DELay	DEL
TYPE	TYPE

In the oscilloscope programmer's documentation, the short form of a command is indicated by uppercase characters.

Programs written in long form are easily read and are almost self-documenting. The short form syntax conserves the amount of controller memory needed for program storage and reduces I/O activity.

Simple Command Headers

Simple command headers contain a single mnemonic. :AUToscale and :DIGitize are examples of simple command headers typically used in the oscilloscope. The syntax is:

```
<program mnemonic><terminator>
```

Simple command headers must occur at the beginning of a program message; if not, they must be preceded by a colon.

When program data must be included with the simple command header (for example, :DIGitize CHANnel1), white space is added to separate the data from the header. The syntax is:

```
<program mnemonic><separator><program data><terminator>
```

Compound Command Headers

Compound command headers are a combination of two or more program mnemonics. The first mnemonic selects the subsystem, and the second mnemonic selects the function within that subsystem. The mnemonics within the compound message are separated by colons. For example, to execute a single function within a subsystem:

```
:<subsystem>:<function><separator><program data><terminator>
```

For example, :CHANnel1:BWLimit ON

Common Command Headers

Common command headers control IEEE 488.2 functions within the instrument (such as clear status). Their syntax is:

```
*<command header><terminator>
```

No space or separator is allowed between the asterisk (*) and the command header. *CLS is an example of a common command header.

Program Data Syntax Rules

Program data is used to convey a parameter information related to the command header. At least one space must separate the command header or query header from the program data.

```
<program mnemonic><separator><data><terminator>
```

When a program mnemonic or query has multiple program data, a comma separates sequential program data.

```
<program mnemonic><separator><data>,<data><terminator>
```

For example, :MEASure:DELay CHANnel1,CHANnel2 has two program data: CHANnel1 and CHANnel2.

Two main types of program data are used in commands: character and numeric.

Character Program Data

Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the :TIMebase:MODE command can be set to normal, 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 commands'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 123)
  - :AALias (see page 125)
  - :COMPlete (see page 126)
  - :COUNt (see page 127)
  - :DAALias (see page 128)
  - :MODE (see page 129)
  - :POINts (see page 130)
  - :SRATe (see page 131)
  - :TYPE (see page 132)
- :AER (Arm Event Register) (see page 89)
- :AUToscale (see page 90)
  - :AMODE (see page 92)
  - :CHANnels (see page 93)
- :BLANk (see page 94)
- :CALibrate (see page 134)
  - :DATE (see page 135)
  - :LABel (see page 136)
  - :STARt (see page 137)
  - :STATus (see page 138)
  - :SWITch (see page 139)
  - :TEMPerature (see page 140)

    - :TIME (see page 141)
- :CDISplay (see page 95)
- :CHANnel<n> (see page 142)
    - :BWLimit (see page 145)
    - :COUPling (see page 146)
    - :DISPlay (see page 147)
    - :IMPedance (see page 148)
    - :INVert (see page 149)
    - :LABel (see page 150)
    - :OFFSet (see page 151)
    - :PROBe (see page 152)
        - :ID (see page 153)
        - :SKEW (see page 154)
        - :STYPe (see page 155)
    - :PROTection (see page 156)
    - :RANGe (see page 157)
    - :SCALe (see page 158)
    - :UNITs (see page 159)
    - :VERNier (see page 160)
- :DIGitize (see page 96)
- :DISPlay (see page 161)
    - :CLEar (see page 163)
    - :DATA (see page 164)
    - :LABel (see page 166)
    - :LABList (see page 167)
    - :PERSistence (see page 168)
    - :SOURce (see page 169)
    - :VECTors (see page 170)
- :EXTernal (see page 171)
    - :BWLimit (see page 173)
    - :IMPedance (see page 174)
    - :PROBe (see page 175)
        - :ID (see page 176)
        - :STYPe (see page 177)
    - :PROTection (see page 178)

    - :RANGe (see page 179)
    - :UNITs (see page 180)
- :FUNCtion (see page 181)
    - :CENTer (see page 183)
    - :DISPlay (see page 184)
    - :OFFSet (see page 185)
    - :OPERation (see page 186)
    - :RANGe (see page 187)
    - :REFerence (see page 188)
    - :SCALe (see page 189)
    - :SOURce (see page 190)
    - :SPAN (see page 191)
    - :WINDow (see page 192)
- :HARDcopy (see page 193)
    - :AREA (see page 195)
    - :APRinter (see page 196)
    - :FACTors (see page 197)
    - :FFEed (see page 198)
    - :INKSaver (see page 199)
    - :PALette (see page 200)
    - [:PRINter]
        - :LIST (see page 201)
    - [:STARt] (see page 202)
- :HWEenable (Hardware Event Enable Register) (see page 98)
- :HWERegister
    - :CONDition (Hardware Event Condition Register) (see page 100)
    - [:EVENt] (Hardware Event Event Register) (see page 102)
- :MARKer (see page 203)
    - :MODE (see page 205)
    - :X1Position (see page 206)
    - :X1Y1source (see page 207)
    - :X2Position (see page 208)
    - :X2Y2source (see page 209)
    - :XDELta (see page 210)
    - :Y1Position (see page 211)

- :Y2Position (see page 212)
- :YDELta (see page 213)

- :MEASure (see page 214)
  - :CLEar (see page 220)
  - :COUNter (see page 221)
  - :DEFine (see page 222)
  - :DELay (see page 225)
  - :DUTYcycle (see page 227)
  - :FALLtime (see page 228)
  - :FREQuency (see page 229)
  - :NWIDth (see page 230)
  - :OVERshoot (see page 231)
  - :PERiod (see page 233)
  - :PHASe (see page 234)
  - :PREShoot (see page 235)
  - :PWIDth (see page 236)
  - :RISetime (see page 237)
  - :SDEViation (see page 238)
  - :SHOW (see page 239)
  - :SOURce (see page 240)
  - :TEDGe (see page 242)
  - :TVALue (see page 244)
  - :VAMPlitude (see page 246)
  - :VAVerage (see page 247)
  - :VBASe (see page 248)
  - :VMAX (see page 249)
  - :VMIN (see page 250)
  - :VPP (see page 251)
  - :VRMS (see page 252)
  - :VTIMe (see page 253)
  - :VTOP (see page 254)
  - :XMAX (see page 255)
  - :XMIN (see page 256)
- :MERGe (see page 104)
- :OPEE (Operation Status Enable Register) (see page 105)

- :OPERegister
  - :CONDition (Operation Status Condition Register) (see page 107)
  - [:EVENt] (Operation Status Event Register) (see page 109)
- :OVLenable (Overload Event Enable Register) (see page 111)
- :OVLRegister (Overload Event Register) (see page 113)
- :RECall
  - :FILename (see page 258)
  - :IMAGe (see page 259)
    - [:STARt] (see page 259)
  - :PWD (see page 260)
  - :SETup (see page 261)
    - [:STARt] (see page 261)
- :RUN (see page 116)
- :SAVE
  - :FILename (see page 264)
  - :IMAGe (see page 265)
    - [:STARt] (see page 265)
    - :AREA (see page 266)
    - :FACTors (see page 267)
    - :FORMat (see page 268)
    - :IGColors (see page 269)
    - :PALette (see page 270)
  - :PWD (see page 271)
  - :SETup (see page 272)
    - [:STARt] (see page 272)
  - :WAVeform (see page 273)
    - [:STARt] (see page 273)
    - :FORMat (see page 274)
    - :LENGth (see page 275)
- :SBUS (see page 276)
  - :CAN
    - :COUNt
      - :ERRor (see page 278)
      - :OVERload (see page 279)
      - :RESet (see page 280)

        - :TOTal (see page 281)
        - :UTILization (see page 282)
    - :DISPlay (see page 283)
    - :IIC
        - :WIDTh (see page 287)
    - :LIN
        - :PARity (see page 285)
    - :MODE (see page 286)
    - :SPI
        - :ASIZe (see page 284)
    - :UART
        - :BASE (see page 288)
        - :COUNt
            - :ERRor (see page 289)
            - :RESet (see page 290)
            - :RXFRames (see page 291)
            - :TXFRames (see page 292)
        - :FRAMing (see page 293)
- :SERial (see page 117)
- :SINGle (see page 118)
- :STATus (see page 119)
- :STOP (see page 120)
- :SYSTem (see page 294)
    - :DATE (see page 295)
    - :DSP (see page 296)
    - :ERRor (see page 297)
    - :LOCK (see page 298)
    - :SETup (see page 299)
    - :TIME (see page 301)
- :TER (Trigger Event Register) (see page 121)
- :TIMebase (see page 302)
    - :MODE (see page 304)
    - :POSition (see page 305)
    - :RANGe (see page 306)
    - :REFerence (see page 307)

        - :SCALe (see page 308)
        - :VERNier (see page 309)
        - :WINDow
            - :POSition (see page 310)
            - :RANGe (see page 311)
            - :SCALe (see page 312)
    - :TRIGger (see page 313)
        - :HFReject (see page 317)
        - :HOLDoff (see page 318)
        - :MODE (see page 319)
        - :NREJect (see page 320)
        - :PATTern (see page 321)
        - :SWEep (see page 323)
        - :CAN (see page 324)
            - :ACKNowledge (see page 495)
            - :PATTern
                - :DATA (see page 326)
                    - :LENGth (see page 327)
                - :ID (see page 328)
                    - :MODE (see page 329)
            - :SAMPlepoint (see page 330)
            - :SIGNal
                - :BAUDrate (see page 331)
                - :DEFinition (see page 496)
            - :SOURce (see page 332)
            - :TRIGger (see page 333)
        - :DURation (see page 335)
            - :GREaterthan (see page 336)
            - :LESSthan (see page 337)
            - :PATTern (see page 338)
            - :QUALifier (see page 339)
            - :RANGe (see page 340)
        - [:EDGE] (see page 341)
            - :COUPling (see page 342)
            - :LEVel (see page 343)

        - :REJect (see page 344)
        - :SLOPe (see page 345)
        - :SOURce (see page 346)
    - :GLITch (see page 347)
        - :GREaterthan (see page 348)
        - :LESSthan (see page 349)
        - :LEVel (see page 350)
        - :POLarity (see page 351)
        - :QUALifier (see page 352)
        - :RANGe (see page 353)
        - :SOURce (see page 354)
    - :HFReject (see page 317)
    - :HOLDoff (see page 318)
    - :IIC (see page 355)
        - :PATTern
            - :ADDRess (see page 356)
            - :DATA (see page 357)
            - :DATa2 (see page 358)
        - :SOURce
            - :CLOCk (see page 359)
            - :DATA (see page 360)
        - :TRIGger
            - :QUALifier (see page 361)
            - [:TYPE] (see page 362)
    - :LIN (see page 364)
        - :ID (see page 365)
        - :SAMPlepoint (see page 366)
        - :SIGNal
            - :BAUDrate (see page 367)
            - :DEFinition (see page 497)
        - :SOURce (see page 368)
        - :STANdard (see page 369)
        - :SYNCbreak (see page 370)
        - :TRIGger (see page 371)
    - :MODE (see page 319)

- :NREJect (see page 320)
- :PATTern (see page 321)
- :SPI (see page 372)
  - :CLOCk
    - :SLOPe (see page 373)
    - :TIMeout (see page 374)
  - :FRAMing (see page 375)
  - :PATTern
    - :DATA (see page 376)
    - :WIDTh (see page 377)
  - :SOURce
    - :CLOCk (see page 378)
    - :DATA (see page 379)
    - :FRAMe (see page 380)
- :SWEep (see page 323)
- :TV (see page 381)
  - :LINE (see page 382)
  - :MODE (see page 383)
  - :POLarity (see page 384)
  - :SOURce (see page 385)
  - :STANdard (see page 386)
  - :TVMode (see page 498)
- :UART (see page 387)
  - :BAUDrate (see page 389)
  - :BITorder (see page 390)
  - :BURSt (see page 391)
  - :DATA (see page 392)
  - :IDLE (see page 393)
  - :PARity (see page 394)
  - :QUALifier (see page 396)
  - :POLarity (see page 395)
  - :SOURce
    - :RX (see page 397)
    - :TX (see page 398)
  - :TYPE (see page 399)

    - :WIDTh (see page 400)
- :VIEW (see page 122)
- :WAVeform (see page 401)
  - :BYTeorder (see page 408)
  - :COUNt (see page 409)
  - :DATA (see page 410)
  - :FORMat (see page 412)
  - :POINts (see page 413)
    - :MODE (see page 415)
  - :PREamble (see page 417)
  - :SOURce (see page 420)
    - :SUBSource (see page 424)
  - :TYPE (see page 425)
  - :UNSigned (see page 426)
  - :VIEW (see page 427)
  - :XINCrement (see page 428)
  - :XORigin (see page 429)
  - :XREFerence (see page 430)
  - :YINCrement (see page 431)
  - :YORigin (see page 432)
  - :YREFerence (see page 433)

Common Commands (IEEE 488.2)

- *CLS (see page 65)
- *ESE (see page 66)
- *ESR (see page 68)
- *IDN (see page 70)
- *LRN (see page 71)
- *OPC (see page 72)
- *OPT (see page 73)
- *RCL (see page 74)
- *RST (see page 75)
- *SAV (see page 78)
- *SRE (see page 79)
- *STB (see page 81)
- *TRG (see page 83)
- *TST (see page 84)

- *WAI (see page 85)

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 535). 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><ter
minator>
```

For example:

```
myScope.WriteString ":TIMebase:RANGe 0.5;POSition 0"
```

NOTE

The colon between TIMebase and RANGe is necessary because TIMebase:RANGe is a compound command. The semicolon between the RANGe command and the POSition command is the required program message unit separator. The POSition command does not need TIMebase preceding it because the TIMebase:RANGe command sets the parser to the TIMebase node in the tree.

Example 2: Program Message Terminator Sets Parser Back to Root

```
myScope.WriteString ":TIMebase:REFerence CENTer;POSition 0.00001"
```

or

```
myScope.WriteString ":TIMebase:REFerence CENTer"
myScope.WriteString ":TIMebase:POSition 0.00001"
```

NOTE

In the first line of example 2, the subsystem selector is implied for the POSition command in the compound command. The POSition command must be in the same program message as the REFerence command because the program message terminator places the parser back at the root of the command tree.

A second way to send these commands is by placing TIMebase: before the POSition command as shown in the second part of example 2. The space after POSition is required.

Example 3: Selecting Multiple Subsystems

You can send multiple program commands and program queries for different subsystems on the same line by separating each command with a semicolon. The colon following the semicolon enables you to enter a new subsystem. For example:

```
<program mnemonic><data>;:<program mnemonic><data><terminator>
```

For example:

```
myScope.WriteString ":TIMebase:REFerence CENTer;:DISPlay:VECTors ON"
```

NOTE

The leading colon before DISPlay:VECTors ON tells the parser to go back to the root of the command tree. The parser can then see the DISPlay:VECTors ON command. The space between REFerence and CENter is required; so is the space between VECTors and ON.

Multiple commands may be any combination of compound and simple commands.

Query Return Values

Command headers immediately followed by a question mark (?) are queries. Queries are used to get results of measurements made by the instrument or to find out how the instrument is currently configured.

After receiving a query, the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued.

When read, the answer is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMebase:RANGe? places the current time base setting in the output queue. When using the Agilent VISA COM library in Visual Basic, the controller statements:

```
Dim strQueryResult As String
myScope.WriteString ":TIMebase:RANGe?"
strQueryResult = myScope.ReadString
```

pass the value across the bus to the controller and place it in the variable strQueryResult.

NOTE

Read Query Results Before Sending Another Command. Sending another command or query before reading the result of a query clears the output buffer (the current response) and places a Query INTERRUPTED error in the error queue.

Infinity Representation

The representation of infinity is +9.9E+37. This is also the value returned when a measurement cannot be made.

All Oscilloscope Commands Are Sequential

IEEE 488.2 makes the distinction between sequential and overlapped commands:

- *Sequential commands* finish their task before the execution of the next command starts.
- *Overlapped commands* run concurrently. Commands following an overlapped command may be started before the overlapped command is completed.

All of the oscilloscope commands are sequential.

9
Programming Examples

SICL Example in C 550
VISA Example in C 559
VISA Example in Visual Basic 568
VISA COM Example in Visual Basic 578

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 "gpib0,7" */                       /* GPIB */
/* #define DEVICE_ADDRESS "lan[a-mso6102-90541]:inst0" */    /* LAN */
#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);
   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.
    */
   iprintf(id, "*RST\n");

   /* Write the *IDN? string and send an EOI indicator, then read
    * the response into buf.
   ipromptf(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:SOURCE CHANNEL1\n");

   /* TRIGGER_MODE - Set the trigger mode to, EDGE, GLITch, 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");
}

/*
 * 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.
    */
   iprintf(id, ":RUN\n");
   iprintf(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.
    */
   iprintf(id, ":BLANK CHANNEL1\n");
   iprintf(id, ":VIEW CHANNEL1\n");

   /* TIME_MODE (not executed in this example) - Set the time base
    * mode to MAIN, DELAYED, XY or ROLL.
    */
   iprintf(id, ":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)
{
   /* AQUIRE_TYPE - Sets the acquisition mode.  There are three
    * acquisition types NORMAL, PEAK, or AVERAGE.
    */
   iprintf(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.
    */
   iprintf(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);
fclose (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];
sdiv  = preamble [2] * preamble [4] / 10;
delay = (preamble [2] / 2) * preamble [4] + preamble [5];

/* 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 %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.
    */

   /* Set waveform format to BYTE. */
   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);
   }
```

```
}

/*
 * 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 C

```
/*
 * 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...\n");
}

/*
 * 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.
    */
   viPrintf(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.
    */
   viPrintf(vi, ":TIMEBASE:REFERENCE CENTER\n");

   /* TRIGGER_SOURCE - Selects the channel that actually produces the
    * TV trigger.  Any channel can be selected.
    */
   viPrintf(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.
    */
   viPrintf(vi, ":TRIGGER:MODE EDGE\n");

   /* TRIGGER_EDGE_SLOPE - Set the slope of the edge for the trigger
    * to either POSITIVE or NEGATIVE.
    */
   viPrintf(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.
    */
   viPrintf(vi, ":RUN\n");
   viPrintf(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)
{
    /* AQUIRE_TYPE - Sets the acquisition mode.  There are three
     * acquisition types NORMAL, PEAK, or AVERAGE.
     */
    viPrintf(vi, ":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.
     */
    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.
     */

    /* 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.
     */
    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. */
   vdiv  = 32 * preamble [7];
   off   = preamble [8];
   sdiv  = preamble [2] * preamble [4] / 10;
   delay = (preamble [2] / 2) * preamble [4] + preamble [5];

   /* 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 %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.
    */

   /* Set waveform format to BYTE. */
   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.
    */
   waveform_size = WAVE_DATA_SIZE;
   /* Read waveform data. */
   viScanf(vi, "%#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);
   }
}

/*
 * 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

```
'
' Agilent VISA Example in Visual Basic
' -------------------------------------------------------------------
' This program illustrates most of the commonly-used programming
' features of your Agilent oscilloscope.
' -------------------------------------------------------------------

Option Explicit

Public err As Long    ' Error returned by VISA function calls.
Public drm As Long    ' Session to Default Resource Manager.
Public vi As Long     ' Session to instrument.

' Declare variables to hold numeric values returned by
' viVScanf/viVQueryf.
Public dblQueryResult As Double
Public Const DblArraySize = 20
Public Const ByteArraySize = 5000000
Public retCount As Long
Public dblArray(DblArraySize) As Double
Public byteArray(ByteArraySize) As Byte
Public paramsArray(2) As Long

' Declare fixed length string variable to hold string value returned
' by viVScanf/viVQueryf.
Public strQueryResult As String * 200

'
' 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()

  ' Open the default resource manager session.
  err = viOpenDefaultRM(drm)

  '  Open the session to the resource.
  '  The "GPIB0" parameter is the VISA Interface name to
  '  an GPIB instrument as defined in:
  '    Start->Programs->Agilent IO Libraries->IO Config
  '  Change this name to whatever you have defined for your
  '  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 actuall 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" + vbLf, 0)

  ' Stop the data acquisition.
  ' err = viVPrintf(vi, ":STOP" + vbLf, 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" + vbLf, 0)

  ' Turn channel 1 on.
  ' err = viVPrintf(vi, ":VIEW CHANNEL1" + vbLf, 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" + vbLf, 0)

End Sub

'
' Capture
' ---------------------------------------------------------------------
' We will capture the waveform using the digitize command.
' ---------------------------------------------------------------------

Private Sub Capture()

  ' AQUIRE_TYPE - Sets the acquisition mode, which can be NORMAL,
  ' PEAK, or AVERAGE.
  err = viVPrintf(vi, ":ACQUIRE:TYPE NORMAL" + vbLf, 0)

  ' 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.
  err = viVPrintf(vi, ":ACQUIRE:COMPLETE 100" + vbLf, 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 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.
  '
  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

  ' Unsigned integer bytes.
  err = viVScanf(vi, "%#b\n" + vbLf, paramsArray(0))
  lngSetupStringSize = retCount
```

```
' 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\n" + 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 = " + _
```

```
'             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.
err = viVPrintf(vi, ":WAV:DATA?" + vbLf, 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" + vbLf, 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 " + _
```

```
        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" + vbLf, 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" + vbLf, 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" + vbLf, 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.

' Set the vertical range to 8 volts.
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.

' Set the time range to 0.002 seconds.
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.

' Set reference to center.
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.

' Set the trigger mode to EDGE.
myScope.WriteString ":TRIGGER:MODE EDGE"

' TRIGGER_EDGE_SLOPE - Sets the slope of the edge for the trigger.

' Set the slope to positive.
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.

' Set time base mode to main.
' 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.
' -------------------------------------------------------------------

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

'
' 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()

  On Error GoTo VisaComError

  ' 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
  ' 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 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) _
    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) * _
    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

Symbols

+9.9E+37, infinity representation, 547
+9.9E+37, measurement error, 219

Numerics

0 (zero) values in waveform data, 410
1 (one) values in waveform data, 410

A

AC coupling, trigger edge, 342
AC input coupling for specified channel, 146
acknowledge, 495
ACQuire commands, 123
acquire data, 96, 132
acquire mode on autoscale, 92
acquire reset conditions, 75
acquire sample rate, 131
acquired data points, 130
acquisition anti-alias control, 125
acquisition count, 127
acquisition mode, 123, 129, 425
acquisition type, 123, 132
active printer, 196
add function, 420
address field size, IIC serial decode, 284
address, IIC trigger pattern, 356
AER (Arm Event Register), 89, 107, 109, 523
ALB waveform data format, 274
alphabetical list of commands, 435
amplitude, vertical, 246
analog channel coupling, 146
analog channel display, 147
analog channel impedance, 148
analog channel input, 464
analog channel inversion, 149
analog channel labels, 150, 167
analog channel offset, 151
analog channel range, 157
analog channel scale, 158
analog channel source for glitch, 354
analog channel units, 159
analog probe attenuation, 152
analog probe sensing, 465
analog probe skew, 154, 463
angle brackets, 57
annotate channels, 150
anti-alias control, 125
AREA commands, 436
area for hardcopy print, 195
area for saved image, 266
Arm Event Register (AER), 89, 107, 109, 523
ASCII format, 412
ASCII format for data transfer, 403
ASCII string, quoted, 57
ASCiixy waveform data format, 274
assign channel names, 150
attenuation factor (external trigger) probe, 175
attenuation for oscilloscope probe, 152
AUT option for probe sense, 465, 469
auto trigger sweep mode, 313
automatic measurements constants, 152
automatic probe type detection, 465, 469
autoscale, 90
autoscale acquire mode, 92
autoscale channels, 93
average value measurement, 247
averaging acquisition type, 123, 403

B

bandwidth filter limits, 173
bandwidth filter limits to 20 MHz, 145
base value measurement, 248
basic instrument functions, 63
baud rate, 331, 367, 389
BAUDrate commands, 436
begin acquisition, 96, 116, 118
binary block data, 57, 299, 410
BINary waveform data format, 274
bit order, 390
bit weights, 68
bitmap display, 164
bits in Service Request Enable Register, 80
bits in Standard Event Status Enable
    Register, 67
bits in Status Byte Register, 82
blank, 94
block data, 57, 71, 164, 299
BMP (bitmap) hardcopy format, 474
braces, 56
button disable, 298
BWLimit commands, 436
byte format for data transfer, 403, 412
BYTeorder, 408

C

C, SICL library example, 550
C, VISA library example, 559
CAL PROTECT switch, 134, 139
calculating preshoot of waveform, 235
calculating the waveform overshoot, 231
calibrate, 135, 136, 139, 141
CALibrate commands, 134
calibrate date, 135
calibrate introduction, 134
calibrate label, 136
calibrate start, 137
calibrate status, 138
calibrate switch, 139
calibrate temperature, 140
calibrate time, 141
CAN, 326
CAN acknowledge, 330, 495
CAN baud rate, 331
CAN commands, 436
CAN frame counters, reset, 280
CAN id pattern, 328
CAN signal definition, 496
CAN source, 332
CAN trigger, 327, 333
CAN trigger commands, 324
CAN trigger pattern id mode, 329
capture data, 96
CDISplay, 95
center frequency set, 182, 183
center of screen, 433
center reference, 307
center screen, vertical value at, 185
channel, 122, 150
CHANnel commands, 142, 143
channel coupling, 146
channel display, 147
channel input impedance, 148
channel inversion, 149
channel label, 150, 462
channel labels, 166, 167
channel overload, 156
channel probe ID, 176
channel protection, 156
channel reset conditions, 75
channel selected to produce trigger, 354, 385
channel signal type, 155
channel skew for oscilloscope probe, 154, 463
channel status, 119
channel vernier, 160
channel, stop displaying, 94
channels to autoscale, 93
channels, how autoscale affects, 90
characters to display, 296
classes of input signals, 192
classifications, command, 530
clear, 163
CLEar commands, 437
clear display, 95
clear markers, 220, 479
clear measurement, 220, 479

clear message queue, 65
clear screen, 467
clear status, 65
clear waveform area, 161
clipped high waveform data value, 410
clipped low waveform data value, 410
clock, 359, 373, 374, 378
CLOCk commands, 437
CLS (Clear Status), 65
CME (Command Error) status bit, 67, 69
code, *RST, 77
code, :ACQuire:COMPlete, 126
code, :ACQuire:TYPE, 133
code, :AUToscale, 91
code, :CHANnel:LABel, 150
code, :CHANnel:PROBe, 152
code, :CHANnel:RANGe, 157
code, :DIGitize, 96
code, :DISPlay:DATA, 165
code, :DISPlay:LABel, 166
code, :MEASure:PERiod, 240
code, :MEASure:TEDGe, 243
code, :RUN/:STOP, 116
code, :SYSTem:SETup, 299
code, :TIMebase:DELay, 494
code, :TIMebase:MODE, 304
code, :TIMebase:RANGe, 306
code, :TIMebase:REFerence, 307
code, :TRIGger:MODE, 319
code, :TRIGger:SLOPe, 345
code, :TRIGger:SOURce, 346
code, :VIEW and :BLANk, 122
code, :WAVeform, 420
code, :WAVeform:DATA, 410
code, :WAVeform:POINts, 414
code, :WAVeform:PREamble, 418
code, SICL library example in C, 550
code, VISA COM library example in Visual Basic, 578
code, VISA library example in C, 559
code, VISA library example in Visual Basic, 568
colon, root commands prefixed by, 88
color palette for hardcopy, 200
color palette for image, 270
Comma Separated Values (CSV) hardcopy format, 474
Comma Separated Values (CSV) waveform data format, 274
command classifications, 530
command errors detected in Standard Event Status, 69
command header, 532
command headers, common, 534
command headers, compound, 533
command headers, simple, 533
command strings, valid, 531
command tree, 535
commands by subsystem, 59
commands in alphabetical order, 435
commands quick reference, 23
commands sent over interface, 63
commands, more about, 529
commands, obsolete and discontinued, 457
common (*) commands, 60, 61, 63
common command headers, 534
completion criteria for an acquisition, 126, 127
compound command headers, 533
compound header, 545
computer control example, 550, 559, 568, 578
conditions for external trigger, 171
conditions, reset, 75
configurations, oscilloscope, 71, 74, 78, 299
connect sampled data points, 466
constants for making automatic measurements, 152
constants for scaling display factors, 152
constants for setting trigger levels, 152
copy display, 115
core commands, 530
count, 409
COUNt commands, 438
count values, 127
counter, 221
coupling, 342
COUPling commands, 438
coupling for channels, 146
CSV (Comma Separated Values) hardcopy format, 474
CSV (Comma Separated Values) waveform data format, 274
current oscilloscope configuration, 71, 74, 78, 299
current probe, 159, 180
cursor mode, 205
cursor position, 206, 208, 210, 211, 213
cursor readout, 480, 484, 485
cursor reset conditions, 75
cursor source, 207, 209
cursor time, 480, 484, 485
cursors track measurements, 239
cursors, how autoscale affects, 90
cursors, X1, X2, Y1, Y2, 204
cycle measured, 227
cycle time, 233

D

data, 326, 357, 360, 376, 379, 410
data 2, 358
DATA commands, 438
data conversion, 403
data displayed, 164
data format for transfer, 403
data output order, 408
data pattern length, 327
data pattern width, 377
data point index, 430
data points, 130
data required to fill time buckets, 126
data structures, status reporting, 509
data transfer, 164
data, erasing, 95
data, saving and recalling, 161
DATE commands, 438
date, calibration, 135
date, system, 295
dB versus frequency, 182
DC coupling for edge trigger, 342
DC input coupling for specified channel, 146
dc RMS measured on waveform, 252
DDE (Device Dependent Error) status bit, 67, 69
decision chart, status reporting, 527
default conditions, 75
define channel labels, 150
define glitch trigger, 352
define measurement, 223
define measurement source, 240
define trigger, 321, 336, 337, 338, 340, 353
defined as, 56
definite-length block response data, 57
DELay commands, 439
delay measured to calculate phase, 234
delay measurement, 223
delay measurements, 242
delay parameters for measurement, 225
delay, how autoscale affects, 90
delayed time base, 304, 494
delayed time base mode, how autoscale affects, 90
delayed window horizontal scale, 312
delta time, 480
delta voltage measurement, 489
delta X cursor, 204
delta Y cursor, 204
DeskJet, 472
destination, 169
detecting probe types, 465, 469
device for hardcopy, 472
device-defined error queue clear, 65
differential signal type, 155, 177
differentiate math function, 128, 182, 186, 420
DIFFerentiate source for function, 190
digital channel data, 403
digital channel labels, 167
digital pod, stop displaying, 94
digitize channels, 96
digits, 57
disable anti-alias mode, 128
disable front panel, 298
disable function, 470
disabling calibration, 139
disabling channel display, 147
disabling status register bits, 66, 79
discontinued and obsolete commands, 457
display clear, 163
DISPlay commands, 161, 439
display commands introduction, 161
display connect, 466
display data, 164
display date, 295
display factors scaling, 152
display for channels, 147
display frequency span, 191
display measurements, 218, 239
display persistence, 168

display reference, 305, 307
display reset conditions, 75
display serial number, 117
display source, 169
display vectors, 170
display, clearing, 95
display, oscilloscope, 168, 169, 170, 184, 296
display, serial decode bus, 283
displaying a baseline, 323
displaying unsynchronized signal, 323
driver, printer, 477
duplicate mnemonics, 545
duration, 336, 337, 340
duration for glitch trigger, 348, 349, 353
duration pattern, 338
duration qualifier, trigger, 336, 337, 339
DURation trigger commands, 335
duration triggering, 313
duty cycle measurement, 218, 227

E

edge coupling, 342
edge define, 321
edge fall time, 228
edge parameter for delay measurement, 225
edge preshoot measured, 235
edge rise time, 237
edge slope, 345
edge source, 346
EDGE trigger commands, 341
edge triggering, 313
edges in measurement, 223
ellipsis, 57
enable channel labels, 166
enabling calibration, 139
enabling channel display, 147
enabling status register bits, 66, 79
end of string (EOS) terminator, 532
end of text (EOT) terminator, 532
end or identify (EOI), 532
enter pattern, 321
EOI (end or identify), 532
EOS (end of string) terminator, 532
EOT (end of text) terminator, 532
Epson, 472
equivalent-time acquisition mode, 124, 129
erase data, 95, 163
erase functions, 95
erase measurements, 479
erase screen, 467
ERRor commands, 439
error frame count (CAN), 278
error frame count (UART), 289
error messages, 297, 499
error number, 297
error queue, 297, 520
error, measurement, 218
ESB (Event Status Bit), 80, 82
ESE (Standard Event Status Enable Register), 66, 519
ESR (Standard Event Status Register), 68, 518
event status conditions occurred, 82
Event Status Enable Register (ESE), 66, 519
Event Status Register (ESR), 68, 121, 518
example code, *RST, 77
example code, :ACQuire:COMPlete, 126
example code, :ACQuire:TYPE, 133
example code, :AUToscale, 91
example code, :CHANnel:LABel, 150
example code, :CHANnel:PROBe, 152
example code, :CHANnel:RANGe, 157
example code, :DIGitize, 96
example code, :DISPlay:DATA, 165
example code, :DISPlay:LABel, 166
example code, :MEASure:PERiod, 240
example code, :MEASure:TEDGe, 243
example code, :RUN/:STOP, 116
example code, :SYSTem:SETup, 299
example code, :TIMebase:DELay, 494
example code, :TIMebase:MODE, 304
example code, :TIMebase:RANGe, 306
example code, :TIMebase:REFerence, 307
example code, :TRIGger:MODE, 319
example code, :TRIGger:SLOPe, 345
example code, :TRIGger:SOURce, 346
example code, :VIEW and :BLANk, 122
example code, :WAVeform, 420
example code, :WAVeform:DATA, 410
example code, :WAVeform:POINts, 414
example code, :WAVeform:PREamble, 418
example programs, 3, 549, 550, 559, 568, 578
EXE (Execution Error) status bit, 67, 69
execution error detected in Standard Event Status, 69
exponential notation, 56
external glitch trigger source, 354
external range, 179
external trigger, 171, 174, 175, 346, 468
EXTernal trigger commands, 171
external trigger input impedance, 174, 468
EXTernal trigger level, 343
external trigger overload, 178
external trigger probe attenuation factor, 175
external trigger probe ID, 176
external trigger probe sensing, 469
external trigger protection, 178
external trigger signal type, 177
EXTernal trigger source, 346
external trigger units, 180

F

FACTors commands, 440
failure, self test, 84
fall time measurement, 218, 228
falling edge, 321
Fast Fourier Transform (FFT) functions, 182, 183, 188, 190, 191, 192
FF values in waveform data, 410
FFT (Fast Fourier Transform) functions, 182, 183, 188, 190, 191, 192
FFT (Fast Fourier Transform) operation, 186, 420
FFT math function, 128
FILename commands, 440
filename for hardcopy, 473
filename for recall, 258
filename for save, 264
filter for frequency reject, 344
filter for high frequency reject, 317
filter for noise reject, 320
filter used to limit bandwidth, 145, 173
filters to Fast Fourier Transforms, 192
fine horizontal adjustment (vernier), 309
fine vertical adjustment (vernier), 160
finish pending device operations, 72
first point displayed, 430
FLATtop window for amplitude measurements, 192
format, 412, 417
FORMat commands, 440
format for block data, 71
format for generic video, 382, 386
format for hardcopy, 471, 474
format for image, 268
format for waveform data, 274
formfeed for hardcopy, 193, 198
frame, 380
frame counters (CAN), error, 278
frame counters (CAN), overload, 279
frame counters (CAN), reset, 280
frame counters (CAN), total, 281
frame counters (UART), error, 289
frame counters (UART), reset, 290
frame counters (UART), Rx frames, 291
frame counters (UART), Tx frames, 292
framing, 375
FRAMing commands, 440
frequency measurement, 218, 229
frequency resolution, 192
frequency span of display, 191
frequency versus dB, 182
front panel mode, 323
front panel Single key, 118
front panel Stop key, 120
front-panel lock, 298
full-scale horizontal time, 306, 311
full-scale vertical axis defined, 187
function, 122, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 470
FUNCtion commands, 181
function memory, 119
function turned on or off, 470
functions, 420
functions, erasing, 95

G

general trigger commands, 316
GENeric, 382, 386
generic video format, 382, 386
glitch duration, 353
glitch qualifier, 352
glitch source, 354
GLITch trigger commands, 347

glitch trigger duration, 348
glitch trigger polarity, 351
glitch trigger source, 348
graphics, 164
graticule area for hardcopy print, 195
graticule area for saved image, 266
graticule colors, invert for hardcopy, 199, 476
graticule colors, invert for image, 269
graticule data, 164
grayscale palette for hardcopy, 200
grayscale palette for image, 270
grayscaling on hardcopy, 475
greater than qualifier, 352
greater than time, 336, 340, 348, 353
GREaterthan commands, 441

H

HANNing window for frequency resolution, 192
hardcopy, 115, 193
HARDcopy commands, 193
hardcopy device, 472
hardcopy factors, 197, 267
hardcopy filename, 473
hardcopy format, 471, 474
hardcopy formfeed, 198
hardcopy grayscale, 475
hardcopy invert graticule colors, 199, 476
hardcopy palette, 200
hardcopy print, area, 195
hardcopy printer driver, 477
hardware event condition register, 100
Hardware Event Condition Register (:HWERegister:CONDition), 100
Hardware Event Condition Register (:OPERegister:CONDition), 525
Hardware Event Enable Register (HWEenable), 98
hardware event event register, 102
Hardware Event Event Register (:HWERegister[:EVENt]), 102, 524
header, 532
high-frequency reject filter, 317, 344
hold until operation complete, 72
holdoff time, 318
holes in waveform data, 410
horizontal adjustment, fine (vernier), 309
horizontal position, 310
horizontal scale, 308, 312
horizontal scaling, 417
horizontal time, 306, 311, 480
HWEenable (Hardware Event Enable Register), 98
HWERegister:CONDition (Hardware Event Condition Register), 100, 525
HWERegister[:EVENt] (Hardware Event Event Register), 102, 524

I

I1080L50HZ, 382, 386
I1080L60HZ, 382, 386
ID commands, 441
id mode, 329
identification number, 70
identification of options, 73
identifier, 328
identifier, LIN, 365
idle until operation complete, 72
IDN (Identification Number), 70
IEEE 488.2 standard, 63
IGColors commands, 441
IIC address, 356
IIC clock, 359
IIC commands, 441
IIC data, 357, 360
IIC data 2, 358
IIC serial decode address field size, 284
IIC trigger commands, 355
IIC trigger qualifier, 361
IIC trigger type, 362
IMAGe commands, 441
image format, 268
image invert graticule colors, 269
image memory, 119, 169
image palette, 270
image, recall, 259
image, save, 265
image, save with inksaver, 269
impedance, 148
IMPedance commands, 442
impedance for external trigger input, 174, 468
infinity representation, 547
initialize, 75
initialize label list, 167
initiate acquisition, 96
inksaver, save image with, 269
input, 174, 468
input coupling for channels, 146
input impedance for channels, 148, 464
input impedance for external trigger, 174, 468
input inversion for specified channel, 149
insert label, 150
installed options identified, 73
instruction header, 532
instrument number, 70
instrument options identified, 73
instrument requests service, 82
instrument serial number, 117
instrument settings, 193
instrument type, 70
integrate math function, 182, 186, 420
INTegrate source for function, 190
INTERN files, 169
internal low-pass filter, 145, 173
introduction to :ACQuire commands, 123
introduction to :CALibrate commands, 134
introduction to :CHANnel commands, 143
introduction to :DISPlay commands, 161
introduction to :EXTernal commands, 171
introduction to :FUNCtion commands, 182
introduction to :HARDcopy commands, 193
introduction to :MARKer commands, 204
introduction to :MEASure commands, 218
introduction to :RECall commands, 257
introduction to :SAVE commands, 263
introduction to :SBUS commands, 277
introduction to :SYSTem commands, 294
introduction to :TIMebase commands, 302
introduction to :TRIGger commands, 313
introduction to :WAVeform commands, 403
introduction to common (*) commands, 63
introduction to root (:) commands, 88
invert graticule colors for hardcopy, 199, 476
invert graticule colors for image, 269
inverting input for channels, 149

K

key disable, 298
key press detected in Standard Event Status Register, 69
knob disable, 298
known state, 75

L

label, 462
LABel commands, 442
label list, 150, 167
labels, 150, 166, 167
labels to store calibration information, 136
labels, specifying, 161
language example program, 550, 559, 568, 578
LaserJet, 472
leakage into peak spectrum, 192
learn string, 71, 299
least significant byte first, 408
left reference, 307
legal values for channel offset, 151
legal values for frequency span, 191
legal values for offset, 185
LENGth commands, 442
length for waveform data, 275
less than qualifier, 352
less than time, 337, 340, 349, 353
LESSthan commands, 442
LEVel commands, 442
level for trigger voltage, 343, 350
LF coupling, 342
license information, 73
limits for line number, 382
LIN acknowledge, 366
LIN baud rate, 367
LIN identifier, 365
LIN serial decode bus parity bits, 285
LIN source, 368
LIN standard, 369
LIN sync break, 370
LIN trigger, 371

LIN trigger commands, 364
LIN trigger definition, 497
line glitch trigger source, 354
line number for TV trigger, 382
line terminator, 56
LINE trigger level, 343
LINE trigger source, 346
list of channel labels, 167
load utilization (CAN), 282
local lockout, 298
lock, 298
lockout message, 298
long form, 532
lower threshold, 233
lower threshold voltage for measurement, 478
lowercase characters in commands, 531
low-frequency reject filter, 344
low-pass filter used to limit bandwidth, 145, 173
LRN (Learn Device Setup), 71
lsbfirst, 408

M

magnitude of occurrence, 244
main sweep range, 310
main time base mode, 304
making measurements, 218
MAN option for probe sense, 465, 469
manual cursor mode, 205
MARKer commands, 203
marker mode, 211
marker position, 212
marker readout, 484, 485
marker set for voltage measurement, 490, 491
marker sets start time, 481
marker time, 480
markers for delta voltage measurement, 489
markers track measurements, 239
markers, command overview, 204
markers, mode, 205
markers, time at start, 485
markers, time at stop, 484
markers, X delta, 210
markers, X1 position, 206
markers, X1Y1 source, 207
markers, X2 position, 208
markers, X2Y2 source, 209
markers, Y delta, 213
markers, Y1 position, 211
markers, Y2 position, 212
mask, 66, 79, 321, 338
master summary status bit, 82
math function, stop displaying, 94
math operations, 182
MAV (Message Available), 65, 80, 82
maximum duration, 336, 337, 349
maximum position, 305
maximum range for delayed window, 311
maximum scale for delayed window, 312
maximum vertical value measurement, 249
maximum vertical value, time of, 255, 482
MEASure commands, 214
measure overshoot, 231
measure period, 233
measure phase between channels, 234
measure preshoot, 235
measure start voltage, 490
measure stop voltage, 491
measure value at a specified time, 253
measure value at top of waveform, 254
measurement error, 218
measurement setup, 218, 240
measurement source, 240
measurements, average value, 247
measurements, base value, 248
measurements, clear, 220, 479
measurements, command overview, 218
measurements, counter, 221
measurements, dc RMS, 252
measurements, definition setup, 223
measurements, delay, 225
measurements, duty cycle, 227
measurements, fall time, 228
measurements, frequency, 229
measurements, how autoscale affects, 90
measurements, lower threshold level, 478
measurements, maximum vertical value, 249
measurements, maximum vertical value, time of, 255, 482
measurements, minimum vertical value, 250
measurements, minimum vertical value, time of, 256, 483
measurements, overshoot, 231
measurements, period, 233
measurements, phase, 234
measurements, preshoot, 235
measurements, pulse width, negative, 230
measurements, pulse width, positive, 236
measurements, resetting, 95
measurements, rise time, 237
measurements, show, 239
measurements, source channel, 240
measurements, standard deviation, 238
measurements, start marker time, 484
measurements, stop marker time, 485
measurements, thresholds, 481
measurements, time between start and stop markers, 480
measurements, time between trigger and edge, 242
measurements, time between trigger and vertical value, 244
measurements, time between trigger and voltage level, 486
measurements, upper threshold value, 488
measurements, vertical amplitude, 246
measurements, vertical peak-to-peak, 251
measurements, voltage difference, 489
memory setup, 78, 299
merge, 104
message available bit, 82
message available bit clear, 65
message displayed, 82
message error, 499
message queue, 517
messages ready, 82
midpoint of thresholds, 233
minimum duration, 336, 337, 340, 348
minimum vertical value measurement, 250
minimum vertical value, time of, 256, 483
mnemonics, duplicate, 545
mode, 129, 132, 205, 304, 383
MODE commands, 444
mode, serial decode, 286
model number, 70
models, oscilloscope, 3
modes for triggering, 319
monochrome palette for image, 270
most significant byte first, 408
move, 182
move cursors, 484, 485
msbfirst, 408
MSG (Message), 80, 82
MSS (Master Summary Status), 82
multiple commands, 545
multiply math function, 182, 186, 420

N

name channels, 150
name list, 167
negative glitch trigger polarity, 351
negative pulse width, 230
negative slope, 345, 373
negative TV trigger polarity, 384
new line (NL) terminator, 56, 532
NL (new line) terminator, 56, 532
noise reject filter, 320
non-core commands, 530
non-interlaced GENeric mode, 386
non-volatile memory, label list, 167
normal acquisition type, 123, 403
normal trigger sweep mode, 313
notices, 2
NR1 number format, 56
NR3 number format, 56
NTSC, 382, 386
NULL string, 296
number format, 56
number of points, 130, 413, 415
number of time buckets, 413, 415
nwidth, 230

O

obsolete and discontinued commands, 457
obsolete commands, 530
occurrence reported by magnitude, 486
offset, 182
OFFSet commands, 444
offset value for channel voltage, 151
offset value for selected function, 185
one values in waveform data, 410
OPC (Operation Complete) command, 72

OPC (Operation Complete) status bit, 67, 69
OPEE (Operation Status Enable Register), 105
operating configuration, 71, 299
operating state, 78
operation complete, 72
operation status condition register, 107
Operation Status Condition Register (:OPERegister:CONDition), 107, 522
operation status conditions occurred, 82
Operation Status Enable Register (OPEE), 105
operation status event register, 109
Operation Status Event Register (:OPERegister[:EVENt]), 109, 521
operation, math, 182
operations for function, 186
OPERegister:CONDition (Operation Status Condition Register), 107, 522
OPERegister[:EVENt] (Operation Status Event Register), 109, 521
OPT (Option Identification), 73
optional syntax terms, 56
options, 73
order of output, 408
oscilloscope external trigger, 171
oscilloscope models, 3
oscilloscope rate, 131
output messages ready, 82
output queue, 72, 516
output queue clear, 65
output sequence, 408
overlapped commands, 548
overload, 156, 178
Overload Event Enable Register (OVL), 111
Overload Event Register (OVLR), 113
overload frame count (CAN), 279
overload protection, 111, 113
overshoot of waveform, 231
overvoltage, 156, 178
OVL (Overload Event Enable Register), 111
OVLR (Overload Event Register), 113
OVLR bit, 102, 107, 109

P

P1080L24HZ, 382, 386
P1080L25HZ, 382, 386
P480L60HZ, 382, 386
P720L60HZ, 382, 386
PAL, 382, 386
PALette commands, 445
palette for hardcopy, 200
palette for image, 270
PAL-M, 382, 386
parameters for delay measurement, 225
parametric measurements, 218
parity, 394
parity bits, LIN serial decode bus, 285
PARity commands, 445
parser, 88, 545
pass, self test, 84
path information, recall, 260
path information, save, 271
pattern, 321, 326, 328, 338, 356, 357, 358, 376
pattern and edge, 321
PATTern commands, 445
pattern duration, 336, 337, 348, 349
pattern length, 327
pattern trigger, 321
pattern triggering, 313
pattern width, 377
peak detect, 132
peak detect acquisition type, 124, 403
peaks, 182
peak-to-peak vertical value measurement, 251
pending operations, 72
percent of waveform overshoot, 231
percent thresholds, 223
period measured to calculate phase, 234
period measurement, 218, 233
persistence, waveform, 161, 168
phase measured between channels, 234
phase measurements, 242
pixel memory, 169
pixel memory, saving display to, 104
PLL Locked bit, 100, 107
pod, 420
pod, stop displaying, 94
points, 130, 413, 415
POINts commands, 445
points in waveform data, 403
polarity, 384, 395
POLarity commands, 446
polarity for glitch trigger, 351
PON (Power On) status bit, 67, 69
position, 208, 305, 310
POSition commands, 446
position cursors, 484, 485
position in delayed view, 310
positive glitch trigger polarity, 351
positive pulse width, 236
positive slope, 345, 373
positive TV trigger polarity, 384
positive width, 236
preamble data, 403, 417
preset conditions, 75
preshoot measured on waveform, 235
previously stored configuration, 74
print command, 115
print job, start, 202
print query, 492
printer, 472
printer driver for hardcopy, 477
printer hardcopy format, 474
printer, active, 196
printing, 193
printing in grayscale, 475
probe, 343
probe attenuation affects channel voltage range, 157
probe attenuation factor (external trigger), 175
probe attenuation factor for selected channel, 152
PROBe commands, 446
probe ID, 153, 176
probe sense for oscilloscope, 465, 469
probe skew value, 154, 463
program data, 532
program data syntax rules, 534
program example, 550, 559, 568, 578
program message, 63
program message syntax, 531
program message terminator, 532
programming examples, 3, 549
protecting against calibration, 139
protection, 111, 113, 156, 178
PROTection commands, 446
pulse width, 230, 236
pulse width duration trigger, 348, 349, 353
pulse width measurement, 218
pulse width trigger, 320
pulse width trigger level, 350
pulse width triggering, 313
PWD commands, 446
pwidth, 236

Q

qualifier, 353
QUALifier commands, 446
qualifier, trigger duration, 336, 337, 339
query error detected in Standard Event Status, 69
query return values, 547
query setup, 193, 204, 218, 299
query subsystem, 182
querying setup, 143
querying the subsystem, 313
queues, clearing, 526
quick reference, commands, 23
quoted ASCII string, 57
QYE (Query Error) status bit, 67, 69

R

range, 182, 311
RANGe commands, 446
range for channels, 157
range for duration trigger, 340
range for external trigger, 179
range for full-scale vertical axis, 187
range for glitch trigger, 353
range for time base, 306
range of offset values, 151
range of reference level values, 188
range qualifier, 352
ranges, value, 57
rate, 131
RCL (Recall), 74
read configuration, 71
read trace memory, 164
readout, 480
real-time acquisition mode, 124, 129
recall, 74, 257, 299
RECall commands, 257

recall filename, 258
recall image, 259
recall setup, 261
recalling and saving data, 161
RECTangular window for transient signals, 192
reference, 182, 307
REFerence commands, 447
reference for time base, 494
reference level, Fast Fourier Transform (FFT) function, 188
registers, 68, 74, 78, 89, 98, 100, 102, 105, 107, 109, 111, 113
registers, clearing, 526
reject filter, 344
reject high frequency, 317
reject noise, 320
remote control example, 550, 559, 568, 578
remove cursor information, 205
remove labels, 166
remove message from display, 296
reorder channels, 90
repetitive acquisitions, 116
report errors, 297
report transition, 242, 244
reporting status, 507
reporting the setup, 313
request service, 82
Request-for-OPC flag clear, 65
reset, 75
RESet commands, 447
reset conditions, 75
reset measurements, 95, 163
resolution of printed copy, 475
restore configurations, 71, 74, 78, 299
restore labels, 166
restore setup, 74
return values, query, 547
returning acquisition type, 132
returning number of data points, 130
right reference, 307
rise time measurement, 218
rise time of positive edge, 237
rising edge, 321
RMS value measurement, 252
roll time base mode, 304
root (:) commands, 86, 88
root level commands, 60
RQL (Request Control) status bit, 67, 69
RQS (Request Service), 82
RST (Reset), 75
rules, tree traversal, 545
rules, truncation, 532
run, 83, 116
Run bit, 107, 109
running configuration, 78, 299
Rx frame count (UART), 291
Rx source, 397

S

sample rate, 131
sampled data, 466
sampled data points, 410
SAMPlepoint commands, 447
SAV (Save), 78
save, 78, 263
SAVE commands, 262
save filename, 264
save image, 265
save image with inksaver, 269
save setup, 272
SAVE TO INTERN, 104
save waveform data, 273
save waveforms to pixel memory, 104
saved image, area, 266
saving and recalling data, 161
SBUS commands, 276
scale, 189, 308, 312
SCALe commands, 448
scale factors output on hardcopy, 197, 267
scale for channels, 158
scale units for channels, 159
scale units for external trigger, 180
scaling display factors, 152
scratch measurements, 479
screen area for hardcopy print, 195
screen area for saved image, 266
screen data, 164
SECAM, 382, 386
seconds per division, 308
select measurement channel, 240
self-test, 84
sensing a channel probe, 465
sensing a external trigger probe, 469
sensitivity of oscilloscope input, 152
sequence triggering, 313
sequential commands, 548
serial clock, 359, 378
serial data, 360, 379
serial decode bus, 277
serial decode bus display, 283
serial decode mode, 286
serial frame, 380
serial number, 117
service request, 82
Service Request Enable Register (SRE), 80, 514
set, 75
set center frequency, 183
set conditions, 90
set cursors, 484, 485
set date, 295
set delay, 90
set thresholds, 90
set time, 301
set time/div, 90
setting display, 184
setting external trigger level, 171
setting impedance for channels, 148
setting inversion for channels, 149
settings, 74, 78
settings, instrument, 193
setup, 124, 143, 161, 182, 193, 299
SETup commands, 448
setup configuration, 74, 78, 299
setup defaults, 75
setup memory, 74
setup reported, 313
setup, recall, 261
setup, save, 272
short form, 3, 532
show channel labels, 166
show measurements, 218, 239
SICL example in C, 550
SIGNal commands, 448
signal type, 155, 177
simple command headers, 533
single acquisition, 118
single-ended signal type, 155, 177
skew, 154, 463
slope, 345, 373
slope (direction) of waveform, 486
SLOPe commands, 449
slope not valid in TV trigger mode, 345
slope parameter for delay measurement, 225
software version, 70
source, 169, 182, 240, 332, 368, 420
SOURce commands, 449
source for function, 190
source for trigger, 346
source for TV trigger, 385
span, 182
span of frequency on display, 191
specify measurement, 240
SPI, 373, 374, 376
SPI commands, 449
SPI decode word width, 287
SPI trigger, 375, 377
SPI trigger clock, 378
SPI trigger commands, 372
SPI trigger data, 379
SPI trigger frame, 380
SRE (Service Request Enable Register), 80, 514
SRQ (Service Request interrupt), 98, 105
STANdard commands, 449
standard deviation measured on waveform, 238
Standard Event Status Enable Register (ESE), 66, 519
Standard Event Status Register (ESR), 68, 518
standard for video, 386
standard, LIN, 369
start acquisition, 83, 96, 116, 118
start and stop edges, 223
STARt commands, 449
start cursor, 484
start measurement, 218
start print job, 202
start time, 353, 484
start time marker, 481
state memory, 78
state of instrument, 71, 299
status, 81, 119, 121
Status Byte Register (STB), 79, 81, 82, 512
STATus commands, 450
status data structure clear, 65
status reporting, 507

STB (Status Byte Register), 79, 81, 82, 512
step size for frequency span, 191
stop, 96, 120
stop acquisition, 120
stop cursor, 485
stop displaying channel, 94
stop displaying math function, 94
stop displaying pod, 94
stop time, 353, 485
storage, 78
store instrument setup, 71, 78
store setup, 78
store waveforms to pixel memory, 104
storing calibration information, 136
string, quoted ASCII, 57
subsource, waveform source, 424
subsystem commands, 60, 545
subtract math function, 182, 186, 420
sweep mode, trigger, 313, 323
sweep speed set to fast to measure fall time, 228
sweep speed set to fast to measure rise time, 237
switch disable, 298
switch, calibration protect, 139
sync break, LIN, 370
syntax elements, 56
syntax rules, program data, 534
syntax, optional terms, 56
syntax, program message, 531
SYSTem commands, 294
system commands, 295, 296, 297, 298, 299, 301
system commands introduction, 294

T

tdelta, 480
tedge, 242
telnet ports 5024 and 5025, 410
temporary message, 296
TER (Trigger Event Register), 121, 515
test, self, 84
text, writing to display, 296
threshold voltage (lower) for measurement, 478
threshold voltage (upper) for measurement, 488
thresholds, 223, 481
thresholds used to measure period, 233
thresholds, how autoscale affects, 90
TIFF image format, 268
time base, 304, 305, 306, 307, 308, 494
time base commands introduction, 302
time base reset conditions, 75
time base window, 310, 311, 312
time between points, 480
time buckets, 126, 127
TIME commands, 450
time delay, 494
time delta, 480
time difference between data points, 428
time duration, 336, 337, 340, 353
time holdoff for trigger, 318
time interval, 242, 244, 480
time interval between trigger and occurrence, 486
time marker sets start time, 481
time per division, 306
time record, 192
time specified, 253
time, calibration, 141
time, start marker, 484
time, stop marker, 485
time, system, 301
time/div, how autoscale affects, 90
time-at-max measurement, 482
time-at-min measurement, 483
TIMebase commands, 302
timebase vernier, 309
time-ordered label list, 167
timeout, 374
timing measurement, 218
title channels, 150
top of waveform value measured, 254
total frame count (CAN), 281
trace memories, how autoscale affects, 90
trace memory, 119, 122
trace memory data, 164
track measurements, 239
trademarks, 2
transfer instrument state, 71, 299
transmit, 164
tree traversal rules, 545
tree, command, 535
TRG (Trigger), 80, 82, 83
trigger (external) input impedance, 174, 468
trigger armed event register, 107, 109
trigger burst, UART, 391
TRIGger CAN commands, 324
trigger channel source, 354, 385
TRIGger commands, 313, 451
TRIGger commands, general, 316
trigger data, UART, 392
trigger duration, 336, 337
TRIGger DURation commands, 335
TRIGger EDGE commands, 341
trigger edge coupling, 342
trigger edge slope, 345
trigger event bit, 121
Trigger Event Register (TER), 515
TRIGger GLITch commands, 347
trigger holdoff, 318
trigger idle, UART, 393
TRIGger IIC commands, 355
trigger level constants, 152
trigger level voltage, 343
TRIGger LIN commands, 364
trigger occurred, 82
trigger pattern, 321, 338
trigger qualifier, 339
trigger qualifier, UART, 396
trigger reset conditions, 75
trigger SPI clock slope, 373
TRIGger SPI commands, 372
trigger status bit, 121
trigger sweep mode, 313
TRIGger TV commands, 381
trigger type, UART, 399
TRIGger UART commands, 387
trigger, CAN, 333
trigger, CAN acknowledge, 495
trigger, CAN pattern data, 326
trigger, CAN pattern data length, 327
trigger, CAN pattern ID, 328
trigger, CAN pattern ID mode, 329
trigger, CAN sample point, 330
trigger, CAN signal baudrate, 331
trigger, CAN signal definition, 496
trigger, CAN source, 332
trigger, duration greater than, 336
trigger, duration less than, 337
trigger, duration pattern, 338
trigger, duration qualifier, 339
trigger, duration range, 340
trigger, edge coupling, 342
trigger, edge level, 343
trigger, edge reject, 344
trigger, edge slope, 345
trigger, edge source, 346
trigger, glitch greater than, 348
trigger, glitch less than, 349
trigger, glitch level, 350
trigger, glitch polarity, 351
trigger, glitch qualifier, 352
trigger, glitch range, 353
trigger, glitch source, 354
trigger, high frequency reject filter, 317
trigger, holdoff, 318
trigger, IIC clock source, 359
trigger, IIC data source, 360
trigger, IIC pattern address, 356
trigger, IIC pattern data, 357
trigger, IIC pattern data 2, 358
trigger, IIC qualifier, 361
trigger, IIC signal baudrate, 367
trigger, IIC type, 362
trigger, LIN, 371
trigger, LIN sample point, 366
trigger, LIN signal definition, 497
trigger, LIN source, 368
trigger, mode, 319
trigger, noise reject filter, 320
trigger, pattern, 321
trigger, SPI clock slope, 373
trigger, SPI clock source, 378
trigger, SPI clock timeout, 374
trigger, SPI data source, 379
trigger, SPI frame source, 380
trigger, SPI framing, 375
trigger, SPI pattern data, 376
trigger, SPI pattern width, 377
trigger, sweep, 323
trigger, TV line, 382
trigger, TV mode, 383, 498
trigger, TV polarity, 384

trigger, TV source, 385
trigger, TV standard, 386
trigger, UART baudrate, 389
trigger, UART bit order, 390
trigger, UART parity, 394
trigger, UART polarity, 395
trigger, UART Rx source, 397
trigger, UART Tx source, 398
trigger, UART width, 400
truncation rules, 532
TST (Self Test), 84
tstart, 484
tstop, 485
turn function on or off, 470
turn off channel, 94
turn off channel labels, 166
turn off cursors, 90
turn off delayed time base mode, 90
turn off digital pod, 94
turn off math function, 94
turn off measurements, 90
turn off trace memories, 90
turn on channel labels, 166
turn on channels, 90
turning channel display on and off, 147
turning off/on function calculation, 184
turning vectors on or off, 466
TV mode, 383, 498
TV trigger commands, 381
TV trigger line number setting, 382
TV trigger mode, 385
TV trigger polarity, 384
TV trigger standard setting, 386
TV triggering, 313
tvmode, 498
Tx data, UART, 424
Tx frame count (UART), 292
Tx source, 398
type, 132, 425
TYPE commands, 453

U

UART baud rate, 389
UART bit order, 390
UART commands, 453
UART frame counters, reset, 290
UART parity, 394
UART polarity, 395
UART Rx source, 397
UART trigger burst, 391
UART trigger commands, 387
UART trigger data, 392
UART trigger idle, 393
UART trigger qualifier, 396
UART trigger type, 399
UART Tx data, 424
UART Tx source, 398
UART width, 400
UNITs commands, 454
units per division, 158, 159, 180, 308
units per division (vertical) for function, 158, 189
unsigned mode, 426
upper threshold, 233
upper threshold voltage for measurement, 488
uppercase characters in commands, 531
URQ (User Request) status bit, 67, 69
user defined channel labels, 150
user event conditions occurred, 82
USR (User Event bit), 80, 82
utilization, CAN bus, 282

V

valid command strings, 531
valid pattern time, 336, 337
value, 244
value measured at base of waveform, 248
value measured at specified time, 253
value measured at top of waveform, 254
value ranges, 57
values required to fill time buckets, 127
vectors, 170
vectors turned on or off, 466
vectors, turning on or off, 161
vernier, channel, 160
vernier, horizontal, 309
vertical adjustment, fine (vernier), 160
vertical amplitude measurement, 246
vertical axis defined by RANGe, 187
vertical axis range for channels, 157
vertical offset for channels, 151
vertical peak-to-peak measured on waveform, 251
vertical scale, 158, 189
vertical scaling, 417
vertical value at center screen, 185
vertical value maximum measured on waveform, 249
vertical value measurements to calculate overshoot, 231
vertical value minimum measured on waveform, 250
video line to trigger on, 382
video standard selection, 386
view, 122, 182, 427
view turns function on or off, 470
VISA COM example in Visual Basic, 578
VISA example in C, 559
VISA example in Visual Basic, 568
Visual Basic, VISA COM example, 578
Visual Basic, VISA example, 568
voltage crossing reported or not found, 486
voltage difference between data points, 431
voltage difference measured, 489
voltage level for active trigger, 343
voltage marker used to measure waveform, 490, 491
voltage offset value for channels, 151
voltage probe, 159, 180
voltage ranges for channels, 157
voltage ranges for external trigger, 179
voltage threshold, 223

W

WAI (Wait To Continue), 85
wait, 85
wait for operation complete, 72
Wait Trig bit, 107, 109
waveform base value measured, 248
WAVeform commands, 401, 455
waveform data, 403
waveform data format, 274
waveform data length, 275
waveform data, save, 273
waveform introduction, 403
waveform maximum vertical value measured, 249
waveform minimum vertical value measured, 250
waveform must cross voltage level to be an occurrence, 486
waveform peak-to-peak vertical value measured, 251
waveform period, 233
waveform persistence, 161
waveform RMS value measured, 252
waveform source channels, 420
waveform source subsource, 424
waveform standard deviation value measured, 238
waveform vertical amplitude, 246
waveform voltage measured at marker, 490, 491
waveform, byte order, 408
waveform, count, 409
waveform, data, 410
waveform, format, 412
waveform, points, 413, 415
waveform, preamble, 417
waveform, source, 420
waveform, type, 425
waveform, unsigned, 426
waveform, view, 427
waveform, X increment, 428
waveform, X origin, 429
waveform, X reference, 430
waveform, Y increment, 431
waveform, Y origin, 432
waveform, Y reference, 433
what's new, 17
width, 353, 400
WIDTh commands, 455
window, 310, 311, 312
window time, 306
window time base mode, 304
windows, 192
windows as filters to Fast Fourier Transforms, 192
windows for Fast Fourier Transform functions, 192
word format, 412
word format for data transfer, 403

word width, SPI decode, 287
write text to display, 296
write trace memory, 164

X

X axis markers, 204
X delta, 210
X1 and X2 cursor value difference, 210
X1 cursor, 204, 206, 207
X2 cursor, 204, 208, 209
X-axis functions, 302
X-increment, 428
X-of-max measurement, 255
X-of-min measurement, 256
X-origin, 429
X-reference, 430
X-Y mode, 302, 304

Y

Y axis markers, 204
Y1 and Y2 cursor value difference, 213
Y1 cursor, 204, 207, 211, 213
Y2 cursor, 204, 209, 212, 213
Y-axis value, 432
Y-increment, 431
Y-origin, 432, 433
Y-reference, 433

Z

zero values in waveform data, 410