

# SDS Series Digital Oscilloscope

Programming Guide

EN11G



## contents

<b>VERSION DECLARATION</b> .....	<b>14</b>
What's New in Version E11G.....	14
What's New in Version E11F .....	15
What's New in Version E11E .....	15
What's New in Version E11D.....	16
What's New in Version E11C.....	16
What's New in Version E11B.....	17
Version E11A at Introduction .....	17
<b>SUPPORTED MODELS</b> .....	<b>18</b>
<b>PROGRAMMING OVERVIEW</b> .....	<b>19</b>
Establishing Communications .....	19
Install NI-VISA .....	19
Connect the Instrument.....	22
Remote Control.....	23
User-defined Programming .....	23
Send SCPI Commands via NI-MAX.....	23
Using SCPI with Telnet.....	23
Using SCPI with Sockets.....	24
Introduction to the SCPI Language .....	25
Command and Query Structure .....	25
Long and Short Form.....	25
Syntax Notation .....	25
Parameter Types .....	26
<b>COMMANDS &amp; QUERIES</b> .....	<b>28</b>
Common (*) Commands.....	29
*IDN .....	30
*OPC.....	30
*RST .....	31
Root(:) Commands .....	32
:AUToset .....	33
:PRINT .....	33
:FORMat:DATA.....	34
ACQUire Commands.....	35
:ACQUire:AMODE .....	36

:ACQuire:CSWeep .....	36
:ACQuire:INTerpolation .....	37
:ACQuire:MMANagement.....	38
:ACQuire:MODE .....	39
:ACQuire:MDEPth .....	40
:ACQuire:NUMACq.....	42
:ACQuire:POINts .....	42
:ACQuire:RESolution.....	43
:ACQuire:SEQuence .....	44
:ACQuire:SEQuence:COUNt.....	45
:ACQuire:SRATe .....	46
:ACQuire:TYPE .....	47
CHANnel Commands .....	48
:CHANnel:REFerence .....	49
:CHANnel<n>:BWLimit.....	50
:CHANnel<n>:COUPling .....	51
:CHANnel<n>:IMPedance .....	52
:CHANnel<n>:INVert .....	53
:CHANnel<n>:LABel.....	54
:CHANnel<n>:LABel:TEXT .....	55
:CHANnel<n>:OFFSet.....	56
:CHANnel<n>:PROBe .....	57
:CHANnel<n>:SCALe .....	58
:CHANnel<n>:SKEW.....	59
:CHANnel<n>:SWITCh.....	60
:CHANnel<n>:UNIT .....	61
:CHANnel<n>:VISible .....	62
COUNter Commands.....	63
:COUNter .....	64
:COUNter:CURRent .....	65
:COUNter:LEVel .....	66
:COUNter:MODE .....	67
:COUNter:SOURce.....	68
:COUNter:STATistics .....	69
:COUNter:STATistics:RESet.....	70
:COUNter:STATistics:VALue.....	71
:COUNter:TOTalizer:GATE.....	72
:COUNter:TOTalizer:GATE:LEVel .....	73

:COUNter:TOTalizer:GATE:SLOPe .....	74
:COUNter:TOTalizer:GATE:TYPE .....	75
:COUNter:TOTalizer:RESet .....	76
:COUNter:TOTalizer:SLOPe .....	76
CURSor Commands .....	77
:CURSor .....	78
:CURSor:TAGStyle .....	79
:CURSor:XREFerence .....	80
:CURSor:YREFerence .....	81
Single Group Cursor Commands .....	82
Multiple Cursors Commands .....	93
DECode Commands .....	136
:DECode .....	137
:DECode:LIST .....	138
:DECode:LIST:LINE .....	139
:DECode:LIST:SCRoll .....	140
:DECode:LIST<n>:RESult .....	141
:DECode:BUS<n> .....	142
:DECode:BUS<n>:COpy .....	143
:DECode:BUS<n>:FORMat .....	144
:DECode:BUS<n>:PROToCol .....	145
:DECode:BUS<n>:RESult .....	146
:DECode:BUS<n>:IIC Commands .....	147
:DECode:BUS<n>:SPI Commandds .....	153
:DECode:BUS<n>:UART Commands .....	168
:DECode:BUS<n>:CAN Commands .....	179
:DECode:BUS<n>:LIN Commands .....	183
:DECode:BUS<n>:FLEXray Commands [Option] .....	187
:DECode:BUS<n>:CANFd Commands [Option] .....	191
:DECode:BUS<n>:IIS Commands [Option] .....	196
:DECode:BUS<n>:M1553 Commands [Option] .....	210
:DECode:BUS<n>:SENT Commands [Option] .....	214
:DECode:BUS<n>:MANChester Commands [Option] .....	224
DIGital Commands [Option] .....	239
:DIGital .....	240
:DIGital:ACTive .....	241
:DIGital:BUS<n>:DISPlay .....	242
:DIGital:BUS<n>:DEFault .....	243

:DIgital:BUS<n>:FORMat .....	244
:DIgital:BUS<n>:MAP .....	245
:DIgital:D<d> .....	246
:DIgital:HEIGht .....	247
:DIgital:LABel<d> .....	248
:DIgital:POINts .....	249
:DIgital:POSition .....	250
:DIgital:SKEW .....	251
:DIgital:SRATe .....	252
:DIgital:THReshold<n> .....	253
DISPlay Commands .....	255
:DISPlay:AXIS .....	256
:DISPlay:AXIS:MODE .....	257
:DISPlay:AXIS:POSition .....	258
:DISPlay:BACKlight .....	259
:DISPlay:CLEar .....	260
:DISPlay:COLor .....	260
:DISPlay:GRATicule .....	261
:DISPlay:GRIDstyle .....	262
:DISPlay:HIDemenu .....	263
:DISPlay:INTensity .....	264
:DISPlay:MENU .....	265
:DISPlay:MENU:HIDE .....	266
:DISPlay:PERsistence .....	267
:DISPlay:TRANsparence .....	268
:DISPlay:TYPE .....	269
DVM Commands .....	270
:DVM .....	271
:DVM:ALARm .....	272
:DVM:ARANge .....	273
:DVM:CURRent .....	274
:DVM:HOLD .....	275
:DVM:MODE .....	276
:DVM:SOURce .....	277
FUNction Commands .....	278
:FUNction:FFTDisplay .....	280
:FUNction:GVALue .....	281
:FUNction<x> .....	282

:FUNction<x>:AVERAge:NUM.....	283
:FUNction<x>:DIFF:DX.....	284
:FUNction<x>:ERES:BITS.....	285
:FUNction<x>:FFT:AUToset.....	286
:FUNction<x>:FFT:HCENter.....	287
:FUNction<x>:FFT:HSCale.....	288
:FUNction<x>:FFT:SPAN.....	289
:FUNction<x>:FFT:LOAD.....	290
:FUNction<x>:FFT:MODE.....	291
:FUNction<x>:FFT:POINts.....	292
:FUNction<x>:FFT:RESET.....	293
:FUNction<x>:FFT:RLEVel.....	294
:FUNction<x>:FFT:SCALe.....	296
:FUNction<x>:FFT:SEARch.....	297
:FUNction<x>:FFT:SEARch:EXCursion.....	298
:FUNction<x>:FFT:SEARch:MARKer<n>.....	299
:FUNction<x>:FFT:SEARch:MARKer<n>:SHOW.....	300
:FUNction<x>:FFT:SEARch:MON.....	301
:FUNction<x>:FFT:SEARch:PORDer.....	302
:FUNction<x>:FFT:SEARch:RESult.....	303
:FUNction<x>:FFT:SEARch:TABLE.....	305
:FUNction<x>:FFT:SEARch:TABLE:DELTA.....	306
:FUNction<x>:FFT:SEARch:TABLE:FREQUency.....	307
:FUNction<x>:FFT:SEARch:THReshold.....	308
:FUNction<x>:FFT:UNIT.....	309
:FUNction<x>:FFT:WINDow.....	310
:FUNction<x>:FILTer:TYPE.....	311
:FUNction<x>:FILTer:HFRequency.....	312
:FUNction<x>:FILTer:LFRequency.....	313
:FUNction<x>:INTegrate:GATE.....	314
:FUNction<x>:INTegrate:OFFSet.....	315
:FUNction<x>:INTErpolate:COEF.....	316
:FUNction<x>:INVert.....	317
:FUNction<x>:LABel.....	318
:FUNction<x>:LABel:TEXT.....	319
:FUNction<x>:MAXHold:SWeeps.....	320
:FUNction<x>:MINHold:SWeeps.....	321
:FUNction<x>:OPERation.....	322

:FUNction<x>:POSition .....	323
:FUNction<x>:SCALe .....	324
:FUNction<x>:SOURce1 .....	325
:FUNction<x>:SOURce2 .....	326
HISTORy Commands .....	327
:HISTORy .....	328
:HISTORy:FRAMe .....	329
:HISTORy:INTERval .....	330
:HISTORy:LIST .....	331
:HISTORy:PLAY .....	332
:HISTORy:TIME .....	333
MEASure Commands .....	334
:MEASure .....	336
:MEASure:ADVanced:CLear .....	336
:MEASure:ADVanced:LINenumber .....	337
:MEASure:ADVanced:P<n> .....	338
:MEASure:ADVanced:P<n>:SOURce1 .....	339
:MEASure:ADVanced:P<n>:SOURce2 .....	340
:MEASure:ADVanced:P<n>:STATistics .....	341
:MEASure:ADVanced:P<n>:SHISTory .....	342
:MEASure:ADVanced:P<n>:TYPE .....	343
:MEASure:ADVanced:P<n>:VALue .....	346
:MEASure:ADVanced:STATistics .....	347
:MEASure:ADVanced:STATistics:AIMLimit .....	348
:MEASure:ADVanced:STATistics:HISTOGRAM .....	349
:MEASure:ADVanced:STATistics:MAXCount .....	350
:MEASure:ADVanced:STATistics:RESet .....	351
:MEASure:ADVanced:STYLe .....	351
:MEASure:ASTRategy .....	352
:MEASure:ASTRategy:BASE .....	353
:MEASure:ASTRategy:TOP .....	354
:MEASure:DITMe<n>:EDGE1 .....	355
:MEASure:DITMe<n>:EDGE2 .....	356
:MEASure:DITMe<n>:SLOPe1 .....	357
:MEASure:DITMe<n>:SLOPe2 .....	358
:MEASure:DITMe<n>:THReshold1 .....	359
:MEASure:DITMe<n>:THReshold2 .....	360
:MEASure:GATE .....	361

:MEASure:GATE:GA .....	362
:MEASure:GATE:GB .....	363
:MEASure:MODE .....	364
:MEASure:RDISplay .....	365
:MEASure:SIMPlE:CLear .....	366
:MEASure:SIMPlE:ITEM.....	366
:MEASure:SIMPlE:SOURce .....	367
:MEASure:SIMPlE:VALue.....	368
:MEASure:THReshold:SOURce.....	369
:MEASure:THReshold:TYPE.....	370
:MEASure:THReshold:ABSolute.....	371
:MEASure:THReshold:PERCent.....	372
MEMory Commands .....	373
:MEMory<m>:HORizontal:POSition .....	374
:MEMory<m>:HORizontal:SCALE.....	375
:MEMory<m>:HORizontal:SYNC .....	376
:MEMory<m>:IMPort .....	377
:MEMory<m>:LABel .....	378
:MEMory<m>:LABel:TEXT .....	379
:MEMory<m>:SWITCh.....	380
:MEMory<m>:VERTical:POSition.....	381
:MEMory<m>:VERTical:SCALE.....	382
MTESt Commands .....	383
:MTESt.....	384
:MTESt:COUNT.....	384
:MTESt:FUNCTion:BUZZer .....	385
:MTESt:FUNCTion:COF .....	386
:MTESt:FUNCTion:FTH.....	387
:MTESt:FUNCTion:SOF .....	388
:MTESt:IDISplay .....	389
:MTESt:MASK:CREate.....	389
:MTESt:MASK:LOAD.....	390
:MTESt:OPERate .....	391
:MTESt:RESet .....	391
:MTESt:SOURce .....	392
:MTESt:TYPE .....	393
RECall Commands .....	394
:RECall:FDEFault .....	395



:RECall:REFErence .....	396
:RECall:SERase .....	397
:RECall:SETup.....	398
REF Commands .....	399
:REF<r>:LABel .....	400
:REF<r>:LABel:TEXT .....	401
:REF<r>:DATA .....	402
:REF<r>:DATA:SOURce.....	403
:REF<r>:DATA:SCALe .....	404
:REF<r>:DATA:POSition.....	405
SAVE Commands .....	406
:SAVE:BINary .....	407
:SAVE:CSV .....	408
:SAVE:DEFault .....	409
:SAVE:IMAGe .....	410
:SAVE:MATLab.....	411
:SAVE:REFerence .....	412
:SAVE:SETup .....	413
SEARch Commands.....	414
:SEARch .....	415
:SEARch:MODE .....	416
:SEARch:COUNt .....	416
:SEARch:EVENT .....	417
:SEARch:COPIY .....	417
:SEARch:EDGE Commands .....	418
:SEARch:SLOPe Commands.....	422
:SEARch:PULSe Commands .....	430
:SEARch:INTerval Commands .....	437
:SEARch:RUNT Commands .....	444
SYSTem Commands .....	452
:SYSTem:BUZZer .....	453
:SYSTem:CLOCK.....	454
:SYSTem:COMMunicate:LAN:GATeway .....	455
:SYSTem:COMMunicate:LAN:IPADdress .....	456
:SYSTem:COMMunicate:LAN:MAC .....	456
:SYSTem:COMMunicate:LAN:SMASK .....	457
:SYSTem:COMMunicate:LAN:TYPE .....	458
:SYSTem:COMMunicate:VNCPort .....	459

:SYSTem:DATE .....	460
:SYSTem:EDUMode .....	461
:SYSTem:LANGuage .....	462
:SYSTem:MENU .....	463
:SYSTem:NStorage .....	464
:SYSTem:NStorage:CONNect .....	465
:SYSTem:NStorage:DISConnect .....	465
:SYSTem:NStorage:STATus .....	465
:SYSTem:PON .....	466
:SYSTem:REBoot .....	466
:SYSTem:REMOte .....	467
:SYSTem:SELFCal .....	468
:SYSTem:SHUTdown .....	468
:SYSTem:SSAVer .....	469
:SYSTem:TIME .....	470
:SYSTem:TOUCh .....	471
TIMEbase Commands .....	472
:TIMEbase:DELAy .....	473
:TIMEbase:REFerence .....	474
:TIMEbase:REFerence:POStion .....	475
:TIMEbase:SCALE .....	476
:TIMEbase:WINDow .....	477
:TIMEbase:WINDow:DELAy .....	478
:TIMEbase:WINDow:SCALE .....	479
TRIGger Commands.....	480
:TRIGger:FREQUency .....	481
:TRIGger:MODE .....	482
:TRIGger:RUN .....	483
:TRIGger:STATus .....	483
:TRIGger:STOP .....	484
:TRIGger:TYPE .....	484
:TRIGger:EDGE Commands .....	485
:TRIGger:SLOPe Commands.....	496
:TRIGger:PULSe Commands.....	510
:TRIGger:VIDeo Commands .....	523
:TRIGger:WINDow Commands.....	534
:TRIGger:INTerval Commands .....	547
:TRIGger:DROPOut Commands.....	560

:TRIGger:RUNT Commands .....	572
:TRIGger:PATtern Commands.....	586
:TRIGger:QUALified Commands.....	597
:TRIGger:DELay Commands .....	607
:TRIGger:NEDGe Commands.....	617
:TRIGger:SHOLd Commands .....	628
:TRIGger:IIC Commands.....	639
:TRIGger:SPI Commands .....	652
:TRIGger:UART Commands.....	669
:TRIGger:CAN Commands.....	684
:TRIGger:LIN Commands.....	693
:TRIGger:FLEXray Commands [Option] .....	707
:TRIGger:CANFd Commands [Option].....	716
:TRIGger:IIS Commands [Option] .....	727
:TRIGger:SENT Commands [Option].....	743
WAVeform Commands .....	746
:WAVeform:BYTeorder.....	747
:WAVeform:SOURce.....	748
:WAVeform:STARt .....	749
:WAVeform:INTerval.....	750
:WAVeform:POINt.....	751
:WAVeform:MAXPoint.....	752
:WAVeform:WIDTh.....	753
:WAVeform:PREamble.....	754
:WAVeform:DATA.....	757
:WAVeform:SEQuence .....	763
WGEN Commands .....	764
ARbWaVe .....	765
BaSic_WaVe.....	767
OUTPut.....	769
SToreList.....	770
SYNC.....	773
VOLTPRT.....	773
METEer Commands .....	774
MMETEer .....	775
READ.....	775
CONFigure Commands.....	776
MEASure Commands.....	784

SENSe Commands .....	793
<b>PROGRAMMING EXAMPLES.....</b>	<b>801</b>
VISA Examples .....	802
VC++ Example .....	802
VB Example .....	808
MATLAB Example .....	813
LabVIEW Example .....	815
C# Example .....	818
Examples of Using Sockets.....	832
Python Example .....	832
C Example .....	835
Common Command Examples.....	837
Read Waveform Data Example.....	837
Read Waveform Data of Digital Example.....	841
Read Waveform Data of FFT Example .....	844
Read Sequence Waveform Data Example .....	847
Screen Dump (PRINT) Example .....	853

## Copyright and Declaration

### Copyright

**SIGLENT TECHNOLOGIES CO., LTD.** All Rights Reserved.

### Trademark Information

**SIGLENT** is the registered trademark of **SIGLENT TECHNOLOGIES CO., LTD.**

### Declaration

- **SIGLENT** products are protected by patent law in and outside of P.R.C.
- **SIGLENT** reserves the right to modify or change parts of or all the specifications or pricing policies at the companys.
- Information in this publication replaces all previously corresponding material.
- Any way of copying, extracting or translating the contents of this manual is not allowed without the permission of **SIGLENT**.

### Product Certification

**SIGLENT** guarantees this product conforms to the national and industrial standards in China and other international standard conformance certifications are in progress.

### Contact Us

If you have any problem or requirement when using our products, please contact **SIGLENT TECHNOLOGIES CO., LTD**

### Headquarters

**SIGLENT Technologies Co., Ltd.**

Add: Blog No.4 & No.5, Antongda Industrial Zone, 3rd Liuxian Road,  
Bao'an District, Shenzhen, 518101, China.

Tel: + 86 755 3688 7876

Fax: + 86 755 3359 1582

Email: [market@siglent.com](mailto:market@siglent.com)

Website: [www.siglent.com/ens](http://www.siglent.com/ens)

## Europe

### **SIGLENT Technologies Germany GmbH**

Add: Staetzlinger Str. 70, 86165 Augsburg, Germany

Tel: +49(0)-821-666 0 111 0

Fax: +49(0)-821-666 0 111 22

Email: [info-eu@siglent.com](mailto:info-eu@siglent.com)

Website [www.siglenteu.com](http://www.siglenteu.com)

## America

### **SIGLENT Technologies NA, Inc**

6557 Cochran Rd Solon, Ohio 44139

Tel: 440-398-5800

Toll Free:877-515-5551

Fax: 440-399-1211

Email: [info@siglentna.com](mailto:info@siglentna.com)

Website: [www.siglentna.com](http://www.siglentna.com)

## Version Declaration

This chapter declares the modifications of command in the most recent release of the programming guide version.

### What's New in Version E11G

This version has corrected the short form of various commands, some command formats, redundant spaces in examples, and errors in Chinese symbols. Updated the waveform reconstruction examples in the chapter Common Command Examples. More detailed descriptions of the new and changed commands appear below.

Command	Represent
:DISPlay:AXIS:POSition	New Command, sets the display position of the vertical axis of the axis label.
:DISPlay:HIDemenu	New Command, hides menu immediately.
:FUNction<x>:FFT:SEARch:MARKer<n>	New Command, sets the frequency of the specified marker.
:FUNction<x>:FFT:SEARch:MARKer<n>:S HOW	New Command, turns on the specified FFT marker.
:FUNction<x>:FFT:SEARch:MON	New Command, sets the FFT marker on peaks or harmonics.
:FUNction<x>:FFT:SEARch:PORDer	New Command, sets the sorting method for FFT peak values.
:FUNction<x>:FFT:SEARch:TABLE	New Command, turns on the FFT search table.
:FUNction<x>:FFT:SEARch:TABLE:DELTA	New Command, sets the display state of increments in the FFT search table.
:FUNction<x>:FFT:SEARch:TABLE:FREQ uency	New Command, sets the display state of frequency in the FFT search table.
:MEASure:ADVanced:P<n>:SHIStory	New Command, returns statistics for the specified advanced measurement item.
:MEASure:DITMe<n>:EDGE1	New Command, sets the starting edge of the specified delta time.
:MEASure:DITMe<n>:EDGE2	New Command, sets the end edge of the specified delta time.
:MEASure:DITMe<n>:SLOPe1	New Command, sets the start edge slope type of the specified delta time.
:MEASure:DITMe<n>:SLOPe2	New Command, sets the end edge slope type of the specified delta time.
:MEASure:DITMe<n>:THReshold1	New Command, sets the threshold type for the start edge of the specified delta time.
:MEASure:DITMe<n>:THReshold2	New Command, sets the threshold type for the end edge of the specified delta time.
:MEASure:RDISplay	New Command, sets the display style of measurement results.
Multiple Cursors Commands	New Commands, different from Single Group Cursor Commands
COUNter:CURRent	New Command, returns the current counter value.

Command	Represent
:COUNter:STATistics:VALue	New Command, returns the the counter statistics.
:DECode:LIST<n>:RESult	New Command, returns the result of the specified decoding list.,Different from the command :DECode:BUS<n>:RESult
:WAVEform:BYTeorder	New Command, sets the byte order of transmitted waveform data words.
:DECode:BUS<n>:PROTOcol :TRIGger:TYPE	Changed, updates parameter ARINC429 to A429
:MEASure:SIMPlE:ITEM :MEASure:ADVanced:P<n>:TYPE	Changed, new parameters ULOWer, DTIMe1, DTIMe2, DTIMe3, and DTIMe4.
:MEASure:ADVanced:P<n>:SOURce1 :MEASure:SIMPlE:SOURce :MEASure:THReshold:SOURce	Changed, new parameters F<x>, ZF<x> and ZM<m>.
:SAVE:IMAGe	Changed, add optional parameter {MON MOFF}

## What's New in Version E11F

New features in version E11F of the software are:

- Support for SDS800X HD, SDS3000X HD.
- Modify the value range of parameter for some commads:  
":CURSor:X1",":CURSor:X2",":CURSor:Y1",":CURSor:Y2",":FUNCTion:GVALue"
- Modify the incorrect parameter enumeration of the commad ":DIGital:BUS<n>:FORMat".
- Add parameters SENT, MANchester, ARINC429, USB20 for the command :DECode:BUS<n>:PROTOCol.
- Add parameters ARINC429 for the command :TRIGger:TYPE.
- Modify the parameter string length of :DIGital: LABel<d> from 7 to 8.

## What's New in Version E11E

New features in version E11E of the software are:

- Support for SDS7000A, SDS1000X HD.
- New support sources for some modules: CURSor, FUNCTion, MEASure, MEMory, SAVE.
- The commad "PRINt" supports for obtaining inverted images.
- New commad in dispay group: ":DISPlay:MENU:HIDE".
- New commad in measure group: ":MEASure:ADVanced:CLEar", ":MEASure:SIMPlE:CLEar".
- Add counter commands.
- Add a read waveform data example in C#.



## What's New in Version E11D

New features in version E11D of the software are:

- Support for SDS6000L.
- Add Qualified, Delay, Nth Edge, Setup/Hold trigger commands.
- New option "FTRIG" for ":TRIGger:MODE" command.
- New commads "TRIGger:FREQuency", "TRIGger:EDGE:IMPedance", "FORMat:DATA".
- Add 1553B, SENT, Manchester decode commands.
- Add bus decoding result query.
- Add FFT Search result query.
- New commands for parameter settings of filter, maxhold, interpolation, average and eres function operators
- Add search commands.
- New commands for axis label settings.
- New commands for horizontal and vertical reference strategy settings.
- New command to get the number of acquired frames.
- Update the save command ":SAVE:BINary <path>" to "SAVE:BINary <path>,<src>".
- Supplementary path description of save commands, the path type can be local, network storage, and udisk.
- Support memory source(Mx), function source(Fx), digital bus(DIGital) for save commands.
- Modify the description of "SAVE:MATLab", which can only be saved in mat format at present.
- The return header of "WAVEform:DATA" shows the number of digits according to the actual data length, instead of the fixed 9 digits.

## What's New in Version E11C

New features in version E11C of the software are:

- Support for SDS2000X HD.
- New vertical resolution command for SDS2000X Plus.
- DVM commands.
- Memory commands.
- Measure cursors commands.
- New measurement item: PSLOPE, NSLOPE, TSR, TSF, THR, THF.
- Update C# example.
- New Read Waveform Data of FFT Example.
- Update Read Waveform Data Example.
- Update Read Sequence Waveform Data Example.

## What's New in Version E11B

New features in version E11B of the software are:

- Measure threshold
- Network storage
- Memory management:Auto,Fixed Memory Depth and Fixed Sampling Rate
- Display menu style: EMBedded|FLOating
- Option for specifying FFT autoset as SPAN|PEAK|NORMAl
- Set FFT span: FUNCtion<x>:FFT:SPAN
- :FUNCtion:INTGate revised to :FUNCtion<x>:INTegrate:GATE
- :FUNCtion:INTGate:GA|GB revised to :FUNCtion:GVALue
- PRINt revised to PRINt?
- Read sequence waveform
- Support reading waveform by piece
- WAV:PRE? and WAV:DATA? return in standard binary block format
- Support for SHS800X/SHS1000X/SDS6000A

## Version E11A at Introduction

Compared with previous versions, this new document redefines the instruction format of each group according to the SCPI specifications and adopts tree-style management. However, not all series models support these instructions, see the next chapter “Supported Models” for details.

## Supported Models

The commands and queries listed in this document can be used for SIGLENTs Digital Oscilloscope Series as shown below. Models are arranged according to their initial release dates.

<b>Model</b>	<b>Version for New Commands</b>
SDS5000X	0.9.0 and later
SDS2000X Plus	1.3.5R3 and later
SDS6000 Pro/ SDS6000A	1.1.7.0 and later
SHS800X/ SHS1000X	1.1.9 and later
SDS2000X HD	1.2.0.2 and later
SDS6000L	1.0.1.0
SDS1000X HD	1.1.0.2
SDS7000A	1.0.7.0
SDS800X HD	1.1.3.1
SDS3000X HD	1.0.3.0

## Programming Overview

This chapter introduces how to build communication between the instrument and the PC. It also introduces how to configure a system for remote instrument control.

Users can remotely control the instrument through USB and LAN interfaces, in combination with National Instruments NI-VISA and programming languages. Through the LAN interface, users can communicate using VXI-11, Sockets and Telnet protocols, depending on the capabilities of the specific instrument.

## Establishing Communications

### Install NI-VISA

USB control requires the National Instruments NI-VISA Library for communications. We also recommend using it for LAN communications for its ease of use, but sockets, telnet, and VXI-11 can also be implemented via LAN connections.

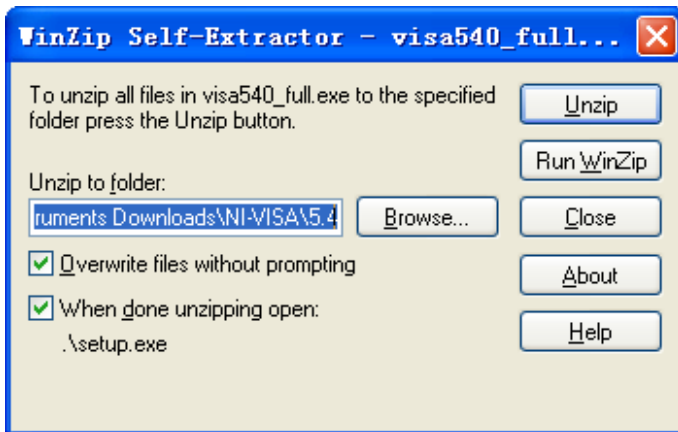
Currently, NI-VISA is packaged in two versions: A full version and a Run-Time Engine version. The full version includes the NI device drivers and a tool named NI MAX which is a user interface to control and test remotely connected devices. The Run-Time Engine is recommended, as it is a much smaller download than the full version and includes the necessary tools for basic communication to instruments.

For example, you can get the NI-VISA 5.4 full version from <http://www.ni.com/download/ni-visa-5.4/4230/en/>.

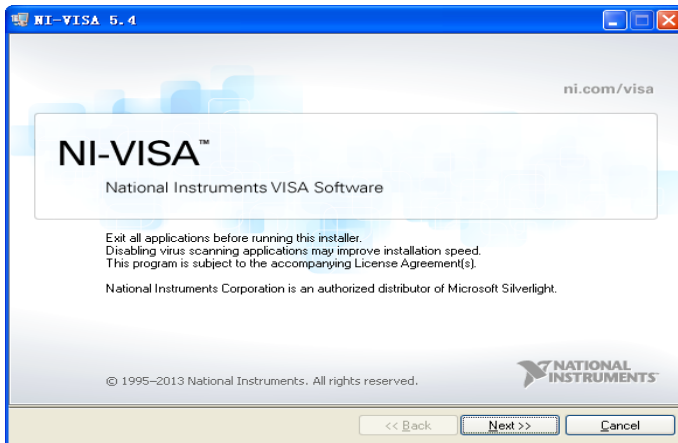
You also can download NI-VISA Run-Time Engine 5.4 to your PC and install it as the default selection. Its installation process is similar to the full version.

After you downloaded the file, follow these steps to install NI-VISA (The full version of NI-VISA 5.4 is used in this example. Newer versions are likely and should be compatible with SIGLENT instrumentation. Download the latest version available for the operating system being used by the controlling computer):

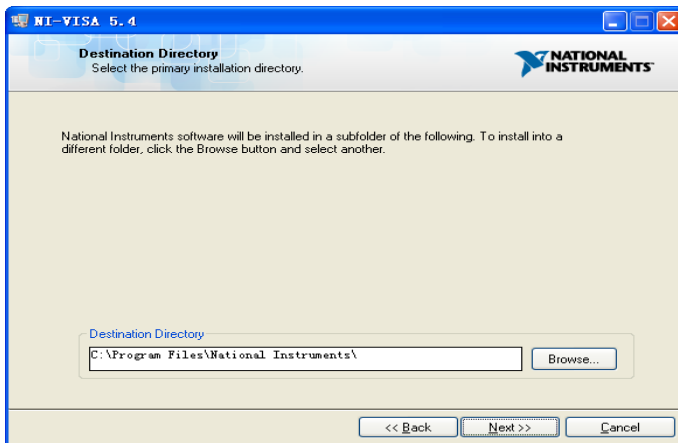
- a. Double click the `visa540_full.exe`, the dialog will be similar to that shown below:



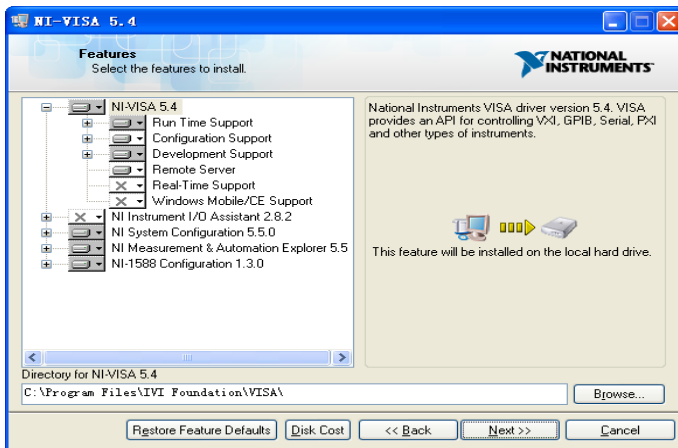
- b. Click Unzip, the installation process will automatically launch after unzipping files. If your computer needs to install .NET Framework 4, it may auto start.



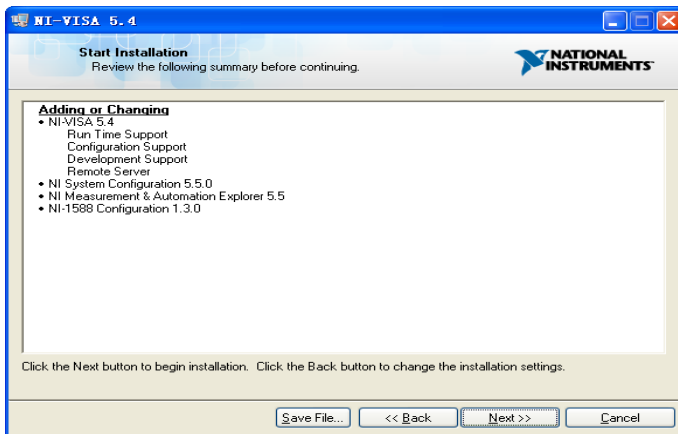
- c. The NI-VISA installing dialog is shown above. Click Next to start the installation process.



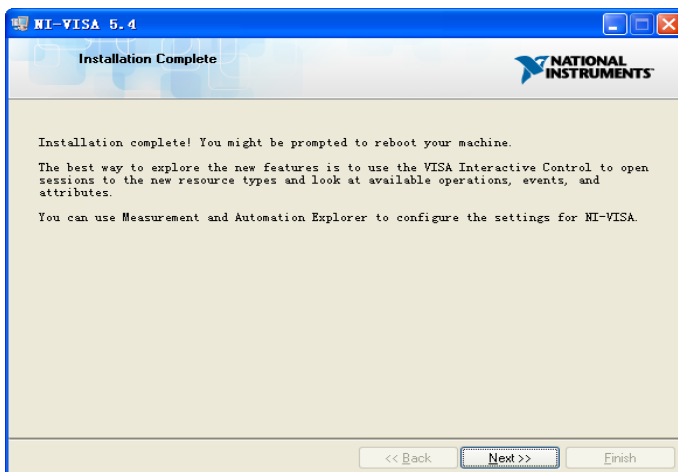
- d. Set the install path. The default path is “C:\Program Files\National Instruments\”, you can change it. Click Next, dialog shown as above.



- e. Click Next twice, in the License Agreement dialog, select the “I accept the above 2 License Agreement(s).”, and click Next, dialog shown as below:



- f. Click Next to begin the installation.

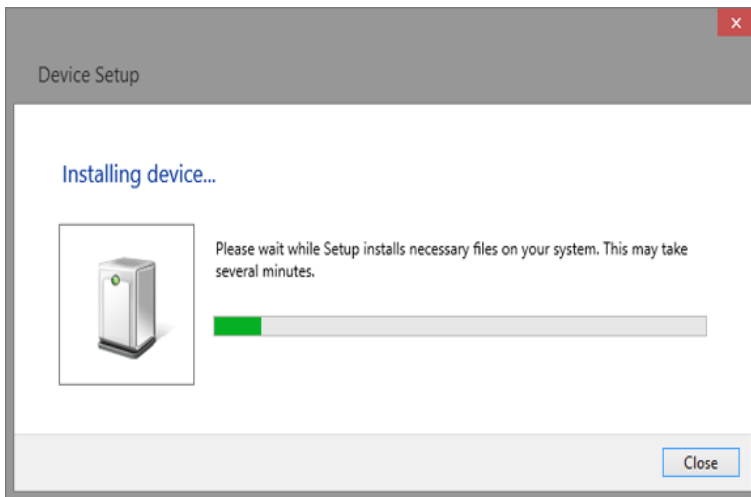


- g. Now the installation is complete. Reboot your PC.

## Connect the Instrument

Depending on the specific model, your oscilloscope may be able to communicate with a PC through the USB or LAN interface.

Connect the instrument and the USB Host interface of the PC using a USB cable. Assuming your PC is already turned on, turn on your oscilloscope, and then the PC will display the “Device Setup” screen as it automatically installs the device driver as shown below.



Wait for the installation to complete and then proceed to the next step.

## Remote Control

### User-defined Programming

Users can use SCPI commands via a computer to program and control the digital oscilloscope. For details, refer to the introductions in "Programming Examples".

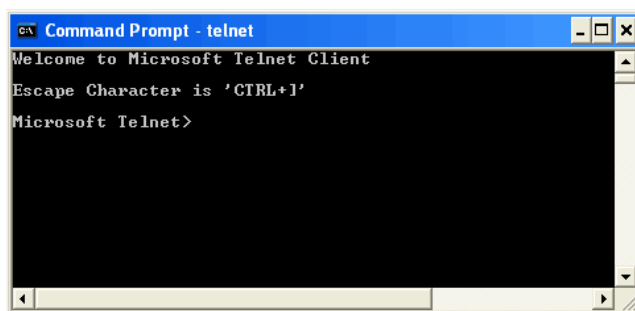
### Send SCPI Commands via NI-MAX

NI-Measurement and Automation eXplorer (NI-MAX) is a program created and maintained by National Instruments. It provides a basic remote control interface for VXI, LAN, USB, GPIB, and Serial communications. It is a utility that enables you to send commands one-at-a-time and also retrieve data from connected devices. It is a great tool for troubleshooting and testing command sequences. The oscilloscopes can be controlled remotely by sending SCPI commands via NI-MAX.

### Using SCPI with Telnet

Telnet provides a means of communicating with the oscilloscopes over a LAN connection. The Telnet protocol sends SCPI commands to the oscilloscopes from a PC and is similar to communicating with the oscilloscopes over USB. It sends and receives information interactively: one command at a time. Windows operating systems use a command prompt style interface for the Telnet client. The steps are as follows:

1. On your PC, click Start > All Programs > Accessories > Command Prompt.
2. At the command prompt, type in telnet.
3. Press the Enter key. The Telnet display screen will be displayed.

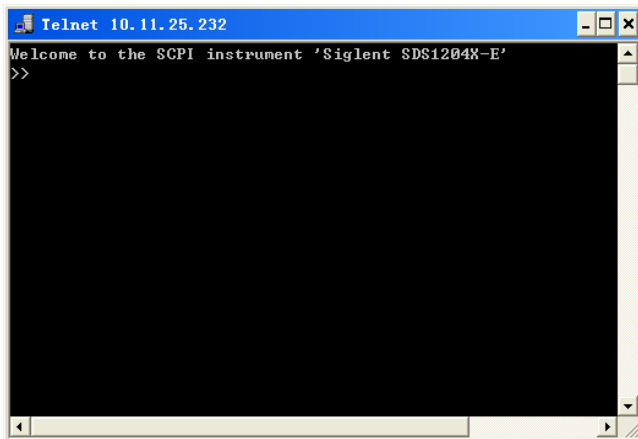


4. At the Telnet command line, type:

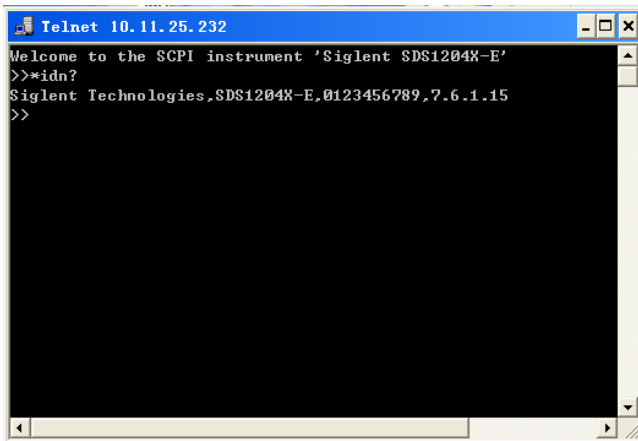
```
open XXX.XXX.XXX.XXX 5024
```

Where XXX.XXX.XXX.XXX is the instrument's IP address and 5024 is the port. You should see a response similar to the following:





- At the SCPI> prompt, input the SCPI commands such as *\*IDN?* to return the company name, model number, serial number, and firmware version number.



- To exit the SCPI> session, press the Ctrl+] keys simultaneously.
- Type *quit* at the prompt or close the Telnet window to close the connection to the instrument and exit Telnet.

### Using SCPI with Sockets

Socket API can be used to control the SDS2000X Plus series via LAN without installing any other libraries. This can reduce the complexity of programming.

<b>SOCKET ADDRESS</b>	IP address+port number
<b>IP ADDRESS</b>	SDS IP address
<b>PORT NUMBER</b>	5025

Please see the section "Examples of Using Sockets" for the details.

## Introduction to the SCPI Language

### Command and Query Structure

Commands consist of set commands and query commands (usually called commands and queries). Commands modify oscilloscope settings or tell the oscilloscope to perform a specific action. Queries cause the oscilloscope to return data and status information. Not all commands have both a set and a query form. Some commands have set only and some have query only.

Commands usually start with a colon [:]. A keyword is separated by a colon (:) followed by optional parameter settings. A question mark (?) is added after the command line to indicate that this function is queried. The command keyword is separated from the first parameter by spaces.

Example:

```
:CHANnel:SCALe <value>  
:CHANnel:SCALe?
```

### Long and Short Form

Each command has both a long and a short form. Note that elsewhere in this document a special notation is employed to differentiate the short form keyword from the long form of the same keyword. The long form of the keyword is shown, with the short form portion shown in uppercase characters, and the rest of the keyword is shown in lowercase characters. If you want to abbreviate, you have to type all the capital letters in the command format.

Example:

```
:CHANnel1:SCALe?  
:CHAN1:SCAL?
```

### Syntax Notation

The following notations are used in the commands:

#### < > (Angle Brackets)

Angle brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command. Parameters are distinguished by enclosing the type name in angle brackets.

#### := (Defined As)

A colon followed by an equals sign separates a placeholder from the description of the type and range of values that may be used in a command instead of the placeholder.

### **{ } (Braces)**

Braces or curly brackets are used to enclose one or more parameters that may be included zero or more times. The vertical bar (|) can be read as “or” and is used to separate alternative parameter options.

### **[ ] (Square Brackets)**

Square brackets are used to enclose a keyword that is optional when programming the command; that is, the instrument shall process the command to have the same effect whether the option node is omitted by the programmer or not.

### **... (Ellipsis)**

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

## **Parameter Types**

### **Enumeration**

Enter these arguments as unquoted text words. Like keywords, enumeration arguments follow the same convention where the portion indicated in uppercase is required and that in lowercase is optional.

### **Numeric**

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

Type	Meaning
<NR1>	Signed integer value
<NR2>	Floating point value without an exponent
<NR3>	Floating point value with an exponent
<bin>	Signed or unsigned integer in binary format

### **Quoted String**

A quoted string is simply a group of ASCII characters enclosed by double quote ("). The following is an example of a quoted string: "This is a quoted string". This documentation represents these arguments as follows: Some commands accept or return data in the form of a quoted string

Type	Meaning
<qstring>	Quoted string of ASCII text

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

1. Use a double quote character to open and close the string.  
Example: "this is a valid string".
2. You can mix quotation marks within a string as long as you follow the previous rule. But cannot include a double quote character within a string by repeating the quote.  
Example: "this is an 'acceptable' string".
3. You cannot include double quotes character within a string by repeating the double quote.  
Example: "here is a "" mark". It will be recognized as "here is a ".
4. Strings can have upper or lower case characters. But the oscilloscope will automatically convert it to uppercase.
5. A carriage return or line feed embedded in a quoted string will be recognized as the string.

Here are some invalid strings:

- "Invalid string argument' (quotes are not of the same type)
- "here is a " " mark" (Duplicate double quotes inside double quotes)

## Commands & Queries

This chapter introduces each command subsystem of the **SIGLENT** Digital Oscilloscope Series command set. The contents of this chapter are shown as below:

- ◆ **Common (\*) Commands**
- ◆ **Root(:) Commands**
- ◆ **ACQUire Commands**
- ◆ **CHANnel Commands**
- ◆ **CURSor Commands**
- ◆ **DECode Commands**
- ◆ **DIGital Commands [Option]**
- ◆ **DISPlay Commands**
- ◆ **DVM Commands**
- ◆ **FUNCTion Commands**
- ◆ **HISTORy Commands**
- ◆ **MEASure Commands**
- ◆ **MEMory Commands**
- ◆ **MTEst Commands**
- ◆ **RECall Commands**
- ◆ **REF Commands**
- ◆ **SAVE Commands**
- ◆ **SYSTEM Commands**
- ◆ **TIMebase Commands**
- ◆ **TRIGger Commands**
- ◆ **WAVEform Commands**
- ◆ **WGEN Commands**

## Common (\*) Commands

The IEEE 488.2 standard defines some general commands for querying the basic information of an instrument or performing common basic operations. These commands usually start with \*, and the command key length is 3 characters.

- ◆ \*IDN
- ◆ \*OPC
- ◆ \*RST

**\*IDN**

**Query**

**DESCRIPTION**

The command query identifies the instrument type and software version. The response consists of four different fields providing information on the manufacturer, the scope model, the serial number and the firmware revision.

**QUERY SYNTAX**

\*IDN?

**RESPONSE FORMAT**

Siglent Technologies,<model>,<serial\_number>,<firmware>

<model>:= The model number of the instrument.

<serial number>:= A 14-character code.

<firmware>:= The software revision of the instrument

**EXAMPLE**

The following command queries the instrument type and software version.

Query message:

*\*IDN?*

Response message:

*Siglent  
Technologies,SDS5104X,SDS5XDAD2R0160,4.6.0.8.7R1*

**\*OPC**

**Query**

**DESCRIPTION**

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

**QUERY SYNTAX**

\*OPC?

**RESPONSE FORMAT**

1

**EXAMPLE**

Query message:

*\*OPC?*

Response message:

*1*

**\*RST****Command****DESCRIPTION**

Resets the oscilloscope to the default configuration, equivalent to the **Default** button on the front panel.

**COMMAND SYNTAX**

\*RST

**EXAMPLE**

The following command resets the oscilloscope.

Command message:

*\*RST*

**RELATED COMMANDS**

:RECall:FDEFault  
:RECall:SETup  
:SAVE:DEFault  
:SAVE:SETup



## Root(:) Commands

The Root commands for querying the basic information of an instrument or performing common basic operations. These commands are only located at the root of the command tree, with no next level and no parameters.

- ◆ **:AUToset**
- ◆ **:PRINt**
- ◆ **:FORMat:DATA**

**:AUToset****Command****DESCRIPTION**

This command attempts to automatically adjust the trigger, vertical, and horizontal controls of the oscilloscope to deliver a usable display of the input signal. Autoset is not recommended for use on low frequency events (< 100 Hz).

**COMMAND SYNTAX**

:AUToset

**EXAMPLE**

Command message:

```
:AUToset  
AUT
```

**:PRINt****Query****DESCRIPTION**

The query captures the screen and returns the data in specified image format.

**QUERY SYNTAX**

:PRINt? <type>[,<format>]

<type>:= {BMP|PNG}

- ◆ BMP selects bitmap format
- ◆ PNG selects Portable Networks Graphics format

<format>:= {NORMal|INVerted}

**RESPONSE FORMAT**

<bin>

Image data in specified image format

**EXAMPLE**

See the code in Screen Dump (PRINt) Example

**:FORMat:DATA****Command/Query****DESCRIPTION**

The command sets the returned precision of the command with data in NR3 format. The current default precision is 3-digits.

The query returns the current precision of the returned data.

**COMMAND SYNTAX**

:FORMat:DATA <option>[,<digit>]

<option>:= {SINGle|DOUBle|CUSTom}

- ◆ SINGle indicates that the single precision type and significant digit is 7.
- ◆ DOUBle indicates that the double precision type and significant digit is 14.
- ◆ CUSTom is user-defined precision, and <digit> need to be set.

<digit>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1,64].

**QUERY SYNTAX**

:FORMat:DATA?

**RESPONSE FORMAT**

CUSTom,<digit>

<digit>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the returned data precision of the command to 5-digits.

Command message:

```
:FORMat:DATA CUSTom,5  
FORM:DATA CUST,5
```

Query message:

```
FORM:DATA?
```

Response message:

```
CUSTom,5
```

## ACQUIRE Commands

The :ACQUIRE subsystem commands control the way in which waveforms are acquired. These commands set the parameters for acquiring and storing data.

- ◆ :ACQUIRE:AMODE
- ◆ :ACQUIRE:CSWEEP
- ◆ :ACQUIRE:INTERPOLATION
- ◆ :ACQUIRE:MMANAGEMENT
- ◆ :ACQUIRE:MODE
- ◆ :ACQUIRE:MDEPTH
- ◆ :ACQUIRE:NUMACQ
- ◆ :ACQUIRE:POINTS
- ◆ :ACQUIRE:RESOLUTION
- ◆ :ACQUIRE:SEQUENCE
- ◆ :ACQUIRE:SEQUENCE:COUNT
- ◆ :ACQUIRE:SRATE
- ◆ :ACQUIRE:TYPE

**:ACQUIRE:AMODE****Command/Query****DESCRIPTION**

The command sets the rate of waveform capture. This command can provide a high-speed waveform capture rate to help capture signal anomalies.

The query returns the current acquisition rate mode.

**COMMAND SYNTAX**

```
:ACQUIRE:AMODE <rate>  
<rate>:= {FAST|SLOW}  
FAST selects fast waveform capture  
SLOW selects slow waveform capture
```

**QUERY SYNTAX**

```
:ACQUIRE:AMODE?
```

**RESPONSE FORMAT**

```
<rate>  
  
<rate>:= {FAST|SLOW}
```

**EXAMPLE**

The following command sets the FAST acquisition rate mode.

Command message:

```
:ACQUIRE:AMODE FAST  
ACQ:AMOD FAST
```

Query message:

```
ACQ:AMOD?
```

Response message:

```
FAST
```

**:ACQUIRE:CSWEEP****Command****DESCRIPTION**

The command clears the sweep and restarts the acquisition. It is equivalent to the Clear Sweeps button on the front panel.

**COMMAND SYNTAX**

```
:ACQUIRE:CSWEEP
```

**EXAMPLE**

The following command clears acquisition sweep.

Command message:

```
:ACQUIRE:CSWEEP  
ACQ:CSW
```

**:ACQuire:INTerpolation****Command/Query****DESCRIPTION**

The command sets the method of interpolation.

The query returns the current method of interpolation.

**COMMAND SYNTAX**

:ACQuire:INTerpolation <state>

<state>:= {ON|OFF}

- ◆ ON selects sinx/x (sinc) interpolation
- ◆ OFF selects linear interpolation

**QUERY SYNTAX**

:ACQuire:INTerpolation?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables sinusoidal interpretation.

Command message:

```
:ACQuire:INTerpolation ON  
ACQ:INT ON
```

Query message:

```
ACQ:INT?
```

Response message:

```
ON
```

**:ACquire:MMANagement****Command/Query****DESCRIPTION**

The command sets the memory mode of the oscilloscope.

The query returns the current memory mode of the oscilloscope.

**COMMAND SYNTAX**

:ACquire:MMANagement <mem\_mode>

<mem\_mode>:= {AUTO|FSRate|FMDepth}

- ◆ AUTO mode maintain the maximum sampling rate, and automatically set the memory depth and sampling rate according to the time base.
- ◆ FSRate mode is Fixed Sampling Rate, maintain the specified sampling rate and automatically set the memory depth according to the time base.
- ◆ FMDepth mode is Fixed Memory Depth, the oscilloscope automatically sets the sampling rate according to the storage depth and time base.

**QUERY SYNTAX**

:ACquire:MMANagement?

**RESPONSE FORMAT**

<mem\_mode>

< mem\_mode>:= {AUTO|FSRate|FMDepth}

**EXAMPLE**

The following command sets the memory mode of the oscilloscope as FMDepth.

Command message:

*:ACquire:MMANagement FMDepth*  
*ACQ:MMAN FMD*

Query message:

*ACQ:MMAN?*

Response message:

*FMDepth*

**:ACQuire:MODE****Command/Query****DESCRIPTION**

The command sets the acquisition mode of the oscilloscope.

The query returns the current acquisition mode of the oscilloscope.

**COMMAND SYNTAX**

:ACQuire:MODE <mode\_type>

<mode\_type>:= {YT|XY|ROLL}

- ◆ YT mode plots amplitude (Y) vs. time (T)
- ◆ XY mode plots channel X vs. channel Y, commonly referred to as a Lissajous curve
- ◆ Roll mode plots amplitude (Y) vs. time (T) as in YT mode, but begins to write the waveforms from the right-hand side of the display. This is similar to a “strip chart” recording and is ideal for slow events that happen a few times/second.

**QUERY SYNTAX**

:ACQuire:MODE?

**RESPONSE FORMAT**

<mode\_type>

<mode\_type>:= {YT|XY|ROLL}

**EXAMPLE**

The following command sets the mode of the oscilloscope as YT.

Command message:

*:ACQuire:MODE YT*

*ACQ:MODE YT*

Query message:

*ACQ:MODE?*

Response message:

*YT*



**:ACQuire:MDEPth**

**Command/Query**

**DESCRIPTION**

The command sets the maximum memory depth.

The query returns the maximum memory depth.

**COMMAND SYNTAX**

:ACQuire:MDEPth <memory\_size>

<memory\_size>:= Varies by model. See the table below for details:

Model	<memory_size>
SDS5000X	Single Channel {250k 1.25M 2.5M 12.5M 25M 125M 250M}
	Dual-Channel {125k 625k 1.25M 6.25M 12.5M 62.5M 125M}
SDS2000X Plus	Single Channel {20k 200k 2M 20M 200M}
	Dual-Channel {10k 100k 1M 10M 100M}
SDS6000 Pro SDS6000A	1G Model Single Channel {1.25k 5k 25k 50k 250k 500k 2.5M 5M 12.5M 125M 250M}
	1G Model Dual-Channel {1.25k 2.5k 12.5k 25k 125k 250k 1.25M 2.5M 12.5M 62.5M 125M}
	2G Model {2.5k 5k 25k 50k 250k 500k 2.5M 5M 12.5M 25M 50M 125M 250M 250M 500M}
SDS6000L	{2.5k 5k 25k 50k 250k 500k 2.5M 5M 12.5M 25M 50M 125M 250M 250M 500M}
SHS800X SHS1000X	Single Channel {12k 120k 1.2M 12M}
	Dual-Channel {6k 60k 600k 6M}
SDS2000X HD	Single Channel {20k 200k 2M 20M 200M}
	Dual-Channel {10k 100k 1M 10M 100M}
SDS800X HD SDS1000X HD	Single Channel {10k 100k 1M 10M 100M}
	Dual-Channel {10k 100k 1M 10M 50M}
	Four-Channel {10k 100k 1M 10M 25M}

SDS7000A	{1k 5k 10k 50k 100k 500k 1M 5M 10M 50M 100M 500M 1G}
SDS3000X HD	Single Channel {2k 10k 20k 100k 200k 1M 10M 20M 100M 200M 400M}
	Dual-Channel {2k 10k 20k 100k 200k 1M 10M 20M 100M 200M}
	Quad-Channel {2k 10k 20k 100k 200k 1M 10M 20M 100M}

**Note:**

- For the definition of single and dual channel mode, please refer to the user manual.
- Turn on digital channels or set the acquisition type to AVERage/ERES or set the acquisition mode to roll, will limit the memory depth.

**QUERY SYNTAX**

:ACquire:MDEPth?

**RESPONSE FORMAT**

&lt;memory\_size&gt;

**EXAMPLE**

The following command sets the memory depth to 125 Mpts for the SDS5000X series.

Command message:

:ACquire:MDEPth 125M

ACQ:MDEP 125M

Query message:

:ACQ:MDEP?

Response message:

125M

**RELATED COMMANDS**

:ACquire:MODE

:ACquire:TYPE

:DIGital

**:ACQUIRE:NUMACq****Query****DESCRIPTION**

The query returns the number of waveform acquisitions that have occurred since starting acquisition. This value is reset to zero when any acquisition, horizontal, or vertical arguments that affect the waveform are changed.

**QUERY SYNTAX**

:ACQUIRE:NUMACq?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following query returns that 350 acquisitions have occurred since starting acquisition.

Query message:

*:ACQUIRE:NUMACq?*  
*ACQ:NUMAC?*

Response message:

*350*

**:ACQUIRE:POINts****Query****DESCRIPTION**

The query returns the number of sampled points of the current waveform on the screen.

**QUERY SYNTAX**

:ACQUIRE:POINts?

**RESPONSE FORMAT**

<point>

<point>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command queries the points of current acquisition.

Query message:

*ACQ:POIN?*

Response message:

*1.25E+08*

**:ACquire:RESolution****Command/Query****DESCRIPTION**

The command sets the ADC resolution for SDS2000X Plus oscilloscope.

The query returns the ADC resolution for SDS2000X Plus oscilloscope.

**COMMAND SYNTAX**

:ACquire:RESolution <bit>

<bit>:= {8Bits|10Bits}

**QUERY SYNTAX**

:ACquire:RESolution?

**RESPONSE FORMAT**

<bit>

<bit>:= {8Bits|10Bits}

**EXAMPLE**

The following command sets the ADC resolution to 10Bits.

Command message:

*:ACquire:RESolution 10Bits*

*ACQ:RES 10B*

Query message:

*ACQ:RES?*

Response message:

*10Bits*

**:ACQUIRE:SEQUENCE****Command/Query****DESCRIPTION**

The command enables or disables sequence acquisition mode.

The query returns whether the current sequence acquisition switch is on or not.

**COMMAND SYNTAX**

:ACQUIRE:SEQUENCE <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:ACQUIRE:SEQUENCE?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on sequence acquisition mode.

Command message:

*:ACQUIRE:SEQUENCE ON*

*ACQ:SEQ ON*

Query message:

*ACQ:SEQ?*

Response message:

*ON*

**RELATED COMMANDS**

:ACQUIRE:SEQUENCE:COUNT

**:ACquire:SEquence:COUNT****Command/Query****DESCRIPTION**

The command sets the number of memory segments to acquire. The maximum number of segments may be limited by the memory depth of your oscilloscope.

The query returns the current count setting.

**COMMAND SYNTAX**

:ACquire:SEquence:COUNT <count>

<count>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value varies from the models and the current timebase, see the user manual for details.

**QUERY SYNTAX**

: ACquire:SEquence:COUNT?

**RESPONSE FORMAT**

<count\_value>

<count\_value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the count of sequence segment as 5.

Command message:

```
:ACquire:SEquence:COUNT 5  
ACQ:SEQ:COUN 5
```

Query message:

```
ACQ:SEQ:COUN?
```

Response message:

```
5
```

**RELATED COMMANDS**

:ACquire:SEquence

**:ACquire:SRATe****Command/Query****DESCRIPTION**

The command set the sampling rate when in the fixed sampling rare mode.

The query returns the current sampling rate.

**COMMAND SYNTAX**

:ACquire:SRATe <rate>

<type>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. If the set value is greater than the settable value, it will automatically match to the settable value.

**QUERY SYNTAX**

:ACquire:SRATe?

**RESPONSE FORMAT**

<sample\_rate>

<sample\_rate>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the current sampling rate.

Command message:

*:ACquire:SRATe 5.00E9*

*ACQ:SRAT 5.00E9*

Query message:

*ACQ:SRAT?*

Response message:

*5.00E+09*

**:ACQuire:TYPE****Command/Query****DESCRIPTION**

The command selects the type of data acquisition that is to take place.

The query returns the current acquisition type.

**COMMAND SYNTAX**

:ACQuire:TYPE <type>

<type>:= {NORMAl|PEAK|AVERAge[,<times>]|ERES[,<bits>]}

<times>:= {4|16|32|64|128|256|512|1024|2048|4096|8192}

<bits>:= {0.5|1.0|1.5|2.0|2.5|3.0|3.5|4.0}

- ◆ NORMAl sets the oscilloscope to normal mode.
- ◆ PEAK sets the oscilloscope to peak detect mode.
- ◆ AVERAge sets the oscilloscope acquisition to averaging mode. You can set the number of averages by sending the command followed by a numeric integer value <times>.
- ◆ ERES sets the oscilloscope to the enhanced resolution mode. This is essentially a digital boxcar filter and is used to reduce noise at slower sweep speeds. You can set the enhanced bits by sending the command followed by the <bits>.

**Note:**

The AVERAge|ERES type is not available when in sequence mode (:ACQuire:SEQuence ON).

**QUERY SYNTAX**

:ACQuire:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {NORMAl|PEAK|AVERAge[,<times>]|ERES[,<bits>]}

<times>:= {4|16|32|64|128|256|512|1024|2048|4096|8192},  
when <type> is AVERAge.

<bits>:= {0.5|1.0|1.5|2.0|2.5|3.0|3.5|4.0} when <type> is ERES.

**EXAMPLE**

The following command sets the acquisition type as AVERAge, and the average number as 16.

Command message:

```
:ACQuire:TYPE AVERAge,16
```

```
ACQ:TYPE AVER,16
```

Query message:

```
ACQ:TYPE?
```

Response message:

```
AVERAge,16
```



## CHANnel Commands

The :CHANnel<n> subsystem commands control the analog channels. Channels are independently programmable for offset, probe, coupling, bandwidth limit, inversion, and more functions. The channel index (1, 2, 3, or 4) specified in the command selects the analog channel that is affected by the command.

- ◆ :CHANnel:REFerence
- ◆ :CHANnel<n>:BWLimit
- ◆ :CHANnel<n>:COUPling
- ◆ :CHANnel<n>:IMPedance
- ◆ :CHANnel<n>:INVert
- ◆ :CHANnel<n>:LABel
- ◆ :CHANnel<n>:LABel:TEXT
- ◆ :CHANnel<n>:OFFSet
- ◆ :CHANnel<n>:PROBe
- ◆ :CHANnel<n>:SCALE
- ◆ :CHANnel<n>:SKEW
- ◆ :CHANnel<n>:SWITCh
- ◆ :CHANnel<n>:UNIT
- ◆ :CHANnel<n>:VISible

**:CHANnel:REFerence****Command/Query****DESCRIPTION**

This command sets the strategy for the offset value change in the vertical direction when the vertical scale is changed.

The query returns the current vertical reference strategy.

**COMMAND SYNTAX**

:CHANnel:REFerence <type>

<type>:= {OFFSet|POSition}

- ◆ OFFset means when the vertical scale is changed, the vertical offset remains fixed. As the vertical scale is changed, the waveform expands/contracts around the main X-axis of the display.
- ◆ POSition means when the vertical scale is changed, the vertical offset remains fixed to the grid position on the display. As the vertical scale is changed, the waveform expands/contracts around the position of the vertical ground position on the display.

**QUERY SYNTAX**

:CHANnel:REFerence?

**RESPONSE FORMAT**

<type>

<type>:= {OFFSet|POSition}

**EXAMPLE**

The following command sets the strategy of the vertical reference to offset.

Command message:

```
:CHANnel:REFerence OFFSet  
CHAN:REF OFFS
```

Query message:

```
CHAN:REF?
```

Response message:

```
OFFSet
```

**:CHANnel<n>:BWLimit****Command/Query****DESCRIPTION**

The command enables or disables the bandwidth-limiting low-pass filter. If the bandwidth filter is on, it will filter the signal to reduce noise and other unwanted high frequency components. When the filter is on, the bandwidth of the specified channel is limited to approximately 20 MHz or 200 MHz.

The query returns the current setting of the low-pass filter.

**COMMAND SYNTAX**

:CHANnel<n>:BWLimit <bwlimit>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<bwlimit>:= {FULL|20M|200M}

- ◆ FULL sets the oscilloscope bandwidth to full.
- ◆ 20M enables the 20 MHz bandwidth filter.
- ◆ 200M enables the 200 MHz bandwidth filter.

**QUERY SYNTAX**

:CHANnel<n>:BWLimit?

**RESPONSE FORMAT**

<bwlimit>

<bwlimit>:= {FULL|20M|200M}

**EXAMPLE**

The following command sets the bandwidth filter of Channel 1 to 20 MHz.

Command message:

*:CHANnel1:BWLimit 20M*  
*CHAN1:BWL 20M*

Query message:

*CHAN1:BWL?*

Response message:

*20M*

**:CHANnel<n>:COUPling****Command/Query****DESCRIPTION**

The command selects the coupling mode of the specified input channel.

The query returns the coupling mode of the specified channel.

**COMMAND SYNTAX**

:CHANnel<n>:COUPling <coupling\_mode>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

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

- ◆ DC sets the channel coupling to DC.
- ◆ AC sets the channel coupling to AC.
- ◆ GND sets the channel coupling to Ground.

**QUERY SYNTAX**

:CHANnel<n>: COUPling?

**RESPONSE FORMAT**

<coupling\_mode>

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

**EXAMPLE**

The following command sets the coupling mode of Channel 1 to AC.

Command message:

*:CHANnel1:COUPling AC*  
*CHAN1:COUP AC*

Query message:

*CHAN1:COUP?*

Response message:

*AC*

**:CHANnel<n>:IMPedance****Command/Query****DESCRIPTION**

The command sets the input impedance of the selected channel. There are two impedance values available, depending on model. They are 1 MOhm and 50.

The query returns the current impedance setting of the selected channel.

**COMMAND SYNTAX**

:CHANnel<n>:IMPedance <impedance>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<impedance>:= {ONEMeg|FIFTy}

- ◆ ONEMeg means 1 Mohm.
- ◆ FIFTy means 50 ohm.

**Note:**

When set to FIFTy, the range of legal values set by the :CHAN<n>:SCAL commands is limited to less than 1 V.

**QUERY SYNTAX**

:CHANnel<n>:IMPedance?

**RESPONSE FORMAT**

<impedance>

<impedance>:= {ONEMeg|FIFTy}

**EXAMPLE**

The following command sets the impedance of Channel 2 to 1 MOhm.

Command message:

```
:CHANnel2:IMPedance ONEMeg  
CHAN2:IMP ONEM
```

Query message:

```
CHAN2:IMP?
```

Response message:

```
ONEMeg
```

**RELATED COMMANDS**

:CHANnel<n>:SCALe

**:CHANnel<n>:INVert****Command/Query****DESCRIPTION**

The command selects whether or not to mathematically invert the input signal for the specified channel. This is a mathematical operation and does not change the polarity of the input signal with reference to ground.

The query returns the current state of the channel inversion.

**COMMAND SYNTAX**

:CHANnel<n>:INVert <state>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {ON|OFF}

- ◆ ON enables channel inversion.
- ◆ Off disables channel inversion.

**QUERY SYNTAX**

:CHANnel<n>:INVert?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command inverts the display of Channel 2.

Command message:

```
:CHANnel2:INVert ON  
CHAN2:INV ON
```

Query message:

```
CHAN2:INV?
```

Response message:

```
ON
```

**:CHANnel<n>:LABel****Command/Query****DESCRIPTION**

The command is to turn the specified channel label on or off.

The query returns the label associated with a particular channel.

**COMMAND SYNTAX**

:CHANnel<n>:LABel <state>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {ON|OFF}

- ◆ ON enables the channel label.
- ◆ OFF disables the channel label.

**QUERY SYNTAX**

:CHANnel<n>:LABel?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the label of Channel 1.

Command message:

*:CHANnel1:LABel ON*  
*CHAN1:LAB ON*

Query message:

*CHAN1:LAB?*

Response message:

*ON*

**:CHANnel<n>:LABel:TEXT****Command/Query****DESCRIPTION**

The command sets the selected 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)

The query returns the current label text of the selected channel.

**COMMAND SYNTAX**

:CHANnel<n>:LABel:TEXT <qstring>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<qstring>:= Quoted string of ASCII text. The length of the string is limited to 20.

**Note:**

All characters will be automatically converted to uppercase.

**QUERY SYNTAX**

:CHANnel<n>:LABel:TEXT?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the label text of Channel 2 to "VOUT".

Command message:

*:CHANnel2:LABel:TEXT "VOUT"*

*CHAN2:LAB:TEXT "VOUT"*

Query message:

*CHAN2:LAB:TEXT?*

Response message:

*"VOUT"*

**RELATED COMMANDS**

:CHANnel<n>:LABel



**:CHANnel<n>:OFFSet****Command/Query****DESCRIPTION**

The command allows adjustment of the vertical offset of the specified input channel. The maximum ranges depend on the fixed sensitivity setting.

The query returns the offset value of the specified channel.

**COMMAND SYNTAX**

:CHANnel<n>:OFFSet <offset\_value>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<offset\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The range of legal values varies with the value set by the :CHANnel<n>:SCALE commands.

**QUERY SYNTAX**

:CHANnel<n>:OFFSet?

**RESPONSE FORMAT**

<offset\_value>

<offset\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the offset of Channel 2 to -3.8 V.

Command message:

```
:CHANnel2:OFFSet -3.8E+00  
CHAN2:OFFS -3.8E+00
```

Query message:

```
CHAN2:OFFS?
```

Response message:

```
-3.8E+00
```

**RELATED COMMANDS**

:CHANnel<n>:SCALE

**:CHANnel<n>:PROBe****Command/Query****DESCRIPTION**

The command specifies the probe attenuation factor for the selected channel. 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.

The query returns the current probe attenuation factor for the selected channel.

**COMMAND SYNTAX**

:CHANnel<n>:PROBe <attenuation>[,<value>]

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<attenuation>:= {DEFault|VALue}

- ◆ DEFault means set to the default value 1X.
- ◆ VALue means set to the <value>.

<value>:= Probe attenuation ratio in NR3 format when <attenuation> is VALue, and the range is [1E-6, 1E6].

**QUERY SYNTAX**

:CHANnel<n>:PROBe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the attenuation factor of Channel 1 to 100:1. To ensure the data matches the true signal voltage values, the physical probe attenuation must match the scope attenuation values for that input channel.

Command message:

```
:CHANnel1:PROBe VALue,1.00E+02
CHAN1:PROB VAL,1.00E+02
```

Query message:

```
CHAN1:PROB?
```

Response message:

```
1.00E+02
```

**RELATED COMMANDS**

```
:CHANnel<n>:SCALe
:CHANnel<n>:OFFSet
```

**:CHANnel<n>:SCALE****Command/Query****DESCRIPTION**

The command sets the vertical sensitivity in Volts/div. If the probe attenuation is changed, the scale value is multiplied by the probe's attenuation factor.

The query returns the current vertical sensitivity of the specified channel.

**COMMAND SYNTAX**

:CHANnel<n>:SCALE <scale>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The range of value varies from the models and the bandwidth of the model. See the data sheet for details.

**QUERY SYNTAX**

:CHANnel<n>:SCALE?

**RESPONSE FORMAT**

<scale>

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The return value is affected by probe.

**EXAMPLE**

The following command sets the vertical sensitivity of Channel 1 to 50 mV/div

Command message:

*:CHANnel1:SCALE 5.00E-02*  
*CHAN1:SCAL 5.00E-02*

Query message:

*CHAN1:SCAL?*

Response message:

*5.00E-02*  
*5.00E-01 (when the probe attenuation ratio is 10:1)*

**RELATED COMMANDS**

:CHANnel<n>:PROBE

**:CHANnel<n>:SKEW**

**Command/Query**

**DESCRIPTION**

The command sets the channel-to-channel skew factor for the specified channel.

The query returns the current probe skew setting for the selected channel.

**COMMAND SYNTAX**

:CHANnel<n>:SKEW <skew\_value>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<skew\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-1.00E-07, 1.00E-07].

**QUERY SYNTAX**

:CHANnel<n>:SKEW?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the skew of Channel 1 to 1.52 ns.

Command message:

*:CHANnel1:SKEW 1.52E-09*  
*CHAN1:SKEW 1.52E-09*

Query message:

*CHAN1:SKEW?*

Response message:

*1.52E-09*

**:CHANnel<n>:SWITCh****Command/Query****DESCRIPTION**

The command turns the display of the specified channel on or off.

The query returns current status of the selected channel.

**COMMAND SYNTAX**

:CHANnel<n>:SWITCh <state>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {OFF|ON}

**QUERY SYNTAX**

:CHANnel<n>:SWITCh?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command displays Channel 1.

Command message:

*:CHANnel1:SWITCh ON*  
*CHAN1:SWIT ON*

Query message:

*CHAN1:SWIT?*

Response message:

*ON*

**:CHANnel<n>:UNIT**

**Command/Query**

**DESCRIPTION**

The command change the unit of input signal of specified channel. There is voltage (V) and current (A) two choice to choose for each channel.

The query returns the current unit of the concerned channel.

**COMMAND SYNTAX**

:CHANnel<n>:UNIT <unit>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<unit>:= {V|A}

**Note:**

The related parameter units are changed to the selected unit after processing this command. This also effects measurement results, cursors value, channel sensitivity, and trigger level.

**QUERY SYNTAX**

:CHANnel<n>:UNIT?

**RESPONSE FORMAT**

<unit>

<unit>:= {V|A}

**EXAMPLE**

The following command sets the unit of Channel 1 to A.

Command message:

*:CHANnel1:UNIT A*  
*CHAN1:UNIT A*

Query message:

*CHAN1:UNIT?*

Response message:

*A*

**:CHANnel<n>:VISible****Command/Query****DESCRIPTION**

The command is used to whether display the waveform of the specified channel or not. Different from the command :CHANnel<n>:SWITCh, it sets the state on the display, and the latter sets the physical switch.

The query returns whether the waveform display function of the selected channel is on or off.

**COMMAND SYNTAX**

:CHANnel<n>:VISible <display\_state>

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<display\_state>:= {ON|OFF}

**QUERY SYNTAX**

:CHANnel<n>:VISible?

**RESPONSE FORMAT**

<display\_state>

<display\_state>:= {ON|OFF}

**EXAMPLE**

The following command sets the display of Channel 2 to ON.

Command message:

*:CHANnel2:VISible ON*  
*CHAN2:VIS ON*

Query message:

*CHAN2:VIS?*

Response message:

*ON*

## COUNTER Commands

The :COUNTER subsystem commands control the counter function.

- ◆ :COUNTER
- ◆ :COUNTER:CURRENT
- ◆ :COUNTER:LEVEL
- ◆ :COUNTER:MODE
- ◆ :COUNTER:SOURCE
- ◆ :COUNTER:STATISTICS
- ◆ :COUNTER:STATISTICS:RESET
- ◆ :COUNTER:STATISTICS:VALUE
- ◆ :COUNTER:TOTALIZER:GATE
- ◆ :COUNTER:TOTALIZER:GATE:LEVEL
- ◆ :COUNTER:TOTALIZER:GATE:SLOPE
- ◆ :COUNTER:TOTALIZER:GATE:TYPE
- ◆ :COUNTER:TOTALIZER:RESET
- ◆ :COUNTER:TOTALIZER:SLOPE



**:COUNter**

**Command/Query**

**DESCRIPTION**

This command sets the switch of the counter function.

The query returns the current state of the counter.

**COMMAND SYNTAX**

:COUNter <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:COUNter?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables counter function.

Command message:

*:COUNter ON*

*COUN ON*

Query message:

*COUN?*

Response message:

*ON*

**:COUNter:CURRent****Query****DESCRIPTION**

The query returns the current counter value.

**QUERY SYNTAX**

:COUNter:CURRent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The counter has 7-digit frequency measurement accuracy. To return a higher precision value, please use the command :FORMat:DATA.

**EXAMPLE**

The following command returns the counter value when the mode of the counter is Frequency.

Query message:

*COUN:CURR?*

Response message:

*1.00E+03*

**:COUNter:LEVel**

**Command/Query**

**DESCRIPTION**

This command specifies the level of the counter.

The query returns the current level of the counter.

**COMMAND SYNTAX**

:COUNter:LEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

Model	Value Range
SDS7000A	$[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$
SDS6000 Pro SDS6000A SDS6000L	$[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$
SDS5000X SDS2000X Plus SDS2000X HD SDS1000X HD SDS800X HD SDS3000X HD	$[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$

**QUERY SYNTAX**

:COUNter:LEVel?

**RESPONSE FORMAT**

<value>:

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the level of the counter to 0.5V.

Command message:

*:COUNter:LEVel 5.00E-01*

*COUN:LEV 5.00E-01*

Query message:

*COUN:LEV?*

Response message:

*5.00E-01*

**:COUNter:MODE****Command/Query****DESCRIPTION**

This command selects the mode of the counter.

The query returns the current mode of the counter.

**COMMAND SYNTAX**

:COUNter: MODE <type>

<type>:= {FREQuency|PERiod|TOTalizer}

- ◆ FREQuency means the average frequency over a set period
- ◆ PERiod means the reciprocal of the average frequency over a set period
- ◆ TOTalizer means the value of cumulative count

**QUERY SYNTAX**

:COUNter:MODE?

**RESPONSE FORMAT**

<type>:= {FREQuency|PERiod|TOTalizer}

**EXAMPLE**

The following command sets the mode of the counter as Frequency.

Command message:

*:COUNter:MODE FREQuency*

*COUN:MODE FREQ*

Query message:

*COUN:MODE?*

Response message:

*FREQuency*

**:COUNter:SOURce****Command/Query****DESCRIPTION**

This command specifies the source of the counter source.

The query returns the current source of the counter.

**COMMAND SYNTAX**

:COUNter: SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:COUNter:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the source of the counter as Channel 1.

Command message:

*:COUNter:SOURce C1*

*COUN:SOUR C1*

Query message:

*COUN:SOUR?*

Response message:

*C1*

**:COUNter:STATistics****Command/Query****DESCRIPTION**

This command sets the switch of the counter statistics function.

The query returns the current state of the counter statistics.

**COMMAND SYNTAX**

:COUNter: STATistics <state>

<state>:= {ON|OFF}

**Note:**

This command can only be used when the counter mode is FREQuency or PERriod.

**QUERY SYNTAX**

:COUNter:STATistics?

**RESPONSE FORMAT**

<state>:

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the switch of the counter statistic to ON.

Command message:

*:COUNter:STATistics ON*  
*COUN:STAT ON*

Query message:

*COUN:STAT?*

Response message:

*ON*

## **:COUNter:STATistics:RESet**

### **Command**

### **DESCRIPTION**

This command resets the statistics results of the counter statistics function.

### **COMMAND SYNTAX**

:COUNter: STATistics:RESet

#### **Note:**

This command can only be used when the counter mode is FREQUency or PERiod.

### **EXAMPLE**

The following command resets the statistics results of the counter statistics function.

Command message:

*:COUNter:STATistics:RESet*

*:COUN:STAT:RES*

### **RELATED COMMANDS**

:COUNter:STATistics

**:COUNter:STATistics:VALue****Query****DESCRIPTION**

The query returns the the counter statistics. This query can only be used when the counter mode is FREQUency or PERiod.

**QUERY SYNTAX**

:COUNter:STATistics:VALue?

**RESPONSE FORMAT**

<current>,<mean>,<min>,<max>,<stdev>,<count>

<current>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. Indicates the current value for statistics.

<mean>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. Indicates the mean value for statistics.

<min>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. Indicates the minimum value for statistics.

<max>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. Indicates the maximum value for statistics.

<stdev>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. Indicates the standard deviation for statistics.

<count>:= Value in NR1 format, including an integer and no decimal point, like 1. Indicates the number of statistics.

**Note:**

- When the switch of the counter statistics is OFF, return "OFF".
- The counter has 7-digit frequency measurement accuracy. To return a higher precision value, please use the command :FORMat:DATA.

**EXAMPLE**

The following command returns the counter statistics when the mode of the counter is Frequency.

Query message:

*COUN:STAT:VAL?*

Response message:

*1.00E+03,1.00E+03,1.00E+03,1.00E+03,1.52E-02,312*



**:COUNter:TOTalizer:GATE**

**Command/Query**

**DESCRIPTION**

The command sets the state of the counter gate when the mode is TOTalizer.

This query returns the current state of the counter gate.

**COMMAND SYNTAX**

:COUNter:TOTalizer:GATE <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:COUNter:TOTalizer:GATE?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the gate setting switch to on.

Command message:

*:COUNter:TOTalizer:GATE ON*  
*COUN:TOT:GATE ON*

Query message:

*COUN:TOT:GATE?*

Response message:

*ON*

**:COUNter:TOTalizer:GATE:LEVel****Command/Query****DESCRIPTION**

The command sets the value of the gate level.

This query returns the value of the gate level.

**COMMAND SYNTAX**

:COUNter:TOTalizer:GATE:LEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:COUNter:TOTalizer:GATE:LEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the style of counter gate to level.

Command message:

*:COUNter:TOTalizer:GATE:LEVel 5.00E-1*

*COUN:TOT:GATE:LEV 5.00E-1*

Query message:

*COUN:TOT:GATE:LEV?*

Response message:

*5.00E-1*

**:COUNter:TOTalizer:GATE:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the gate source when the gate type is AEDGE, and sets the polarity of the gate source when the gate type is LEVEL.

This query returns the slope or polarity of the gate source.

**COMMAND SYNTAX**

:COUNter:TOTalizer:GATE:SLOPe <slope>

<slope>:= {RISing|FALLing}

**QUERY SYNTAX**

:COUNter:TOTalizer:GATE:SLOPe?

**RESPONSE FORMAT**

<slope>

<slope>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the slope of the counter gate to RISing when the gate type is AEDGE.

Command message:

```
:COUNter:TOTalizer:GATE:SLOPe RISing  
COUN:TOT:GATE:SLOP RIS
```

Query message:

```
COUN:TOT:GATE:SLOP?
```

Response message:

```
RISing
```

**:COUNter:TOTalizer:GATE:TYPE****Command/Query****DESCRIPTION**

The command selects the type of the counter gate.

This query returns the current style of the counter gate.

**COMMAND SYNTAX**

:COUNter:TOTalizer:GATE:TYPE <style>

<style>:= {LEVel|AEDGE}

**QUERY SYNTAX**

:COUNter:TOTalizer:GATE:TYPE?

**RESPONSE FORMAT**

<style>

<style>:= {LEVel|AEDGE}

**EXAMPLE**

The following command sets the style of counter gate to level.

Command message:

*:COUNter:TOTalizer:GATE:TYPE LEVel*

*COUN:TOT:GATE:TYPE LEV*

Query message:

*COUN:TOT:GATE:TYPE?*

Response message:

*LEVel*

**:COUNter:TOTalizer:RESet**

**Command**

**DESCRIPTION** This command resets the results of the counter totalizer function.

**COMMAND SYNTAX** :COUNter: TOTalizer:RESet

**EXAMPLE** The following command resets the results of the counter totalizer function.

Command message:  
*:COUNter:TOTalizer:RESet*  
*:COUN:TOT:RES*

**:COUNter:TOTalizer:SLOPe**

**Command/Query**

**DESCRIPTION** The command sets the slope of the counter totalizer source.

This query returns the slope of the counter totalizer source.

**COMMAND SYNTAX** :COUNter: TOTalizer:SLOPe <slope>

<slope>:= {RISing|FALLing}

**QUERY SYNTAX** :COUNter: TOTalizer:SLOPe?

**RESPONSE FORMAT** <slope>

<slope>:= {RISing|FALLing}

**EXAMPLE** The following command sets the slope of counter totalizer source to RISing.

Command message:  
*:COUNter:TOTalizer:SLOPe RISing*  
*COUN:TOT:SLOP RIS*

Query message:  
*COUN:TOT:SLOP?*

Response message:  
*RISing*

## **CURSor Commands**

The CURSor command subsystem commands control the cursor measurement function and allows for manual cursor, tracking cursor, and measurement cursor settings.

The Siglent oscilloscope supports two forms of cursor: single group cursor and multiple cursors. The command systems for the two forms of cursor are different, please use the correct commands according to the cursor form.

- ◆ **:CURSor**
- ◆ **:CURSor:TAGStyle**
- ◆ **:CURSor:XREFerence**
- ◆ **:CURSor:YREFerence**
- ◆ **Single Group Cursor Commands**
- ◆ **Multiple Cursors Commands**

**:CURSor****Command/Query****DESCRIPTION**

The command chooses whether to open the cursor.

This command query returns the current state of the cursor.

**COMMAND SYNTAX**

:CURSor <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:CURSor?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables cursor function.

Command message:

*:CURSor ON*

*CURS ON*

Query message:

*CURS?*

Response message:

*ON*

**:CURSor:TAGStyle****Command/Query****DESCRIPTION**

The command selects the tag type of the cursor value.

The query returns the current tag type of cursor value.

**COMMAND SYNTAX**

:CURSor:TAGStyle <type>

<type>:= {FIXed|FOLLOwing}

**QUERY SYNTAX**

:CURSor:TAGStyle?

**RESPONSE FORMAT**

<type>

<type>:= {FIXed|FOLLOwing}

**EXAMPLE**

The following command sets the tag type of cursor value to FIXed.

Command message:

```
:CURSor:TAGStyle FIXed  
CURS:TAGS FIXed
```

Query message:

```
CURS:TAGS?
```

Response message:

```
FIXed
```



**:CURSor:XREFerence****Command/Query****DESCRIPTION**

This command specifies the expansion strategy around the cursor X.

The query returns the expansion strategy of the cursor X.

**COMMAND SYNTAX**

:CURSor:XREFerence <type>

<type>:= {DELay|POSition}

- ◆ DELay means that the cursor value is fixed, and the on-screen cursor position changes for different timebase values.
- ◆ POSition means that the cursor position is fixed, and does not change at any time. Timebase changes cause an expansion or contraction of the waveforms around the cursor position.

**QUERY SYNTAX**

:CURSor:XREFerence?

**RESPONSE FORMAT**

<type>

< type >:= {DELay|POSition}

**EXAMPLE**

The following command sets the type of the X cursor reference to delay.

Command message:

```
:CURSor:XREFerence DELay  
CURS:XREF DEL
```

Query message:

```
CURS:XREF?
```

Response message:

```
DELay
```

**:CURSor:YREFerence****Command/Query****DESCRIPTION**

This command specifies the expansion strategy of the Y cursor.

The query returns the expansion strategy of the Y cursor.

**COMMAND SYNTAX**

:CURSor:YREFerence <type>

<type>:= {OFFSet|POSition}

- ◆ OFFSet means that the cursor value is fixed, and the cursor position moves with vertical scale changes. The cursors expand or contract if the vertical scale changes.
- ◆ POSition means that the cursor position is fixed, and does not change at any time.

**QUERY SYNTAX**

:CURSor:YREFerence?

**RESPONSE FORMAT**

<type>

<type>:= {OFFSet|POSition}

**EXAMPLE**

The following command sets the type of the Y cursor reference to offset.

Command message:

```
:CURSor:YREFerence OFFSet  
CURS:YREF OFFS
```

Query message:

```
CURS:YREF?
```

Response message:

```
OFFSet
```

## Single Group Cursor Commands

The single group cursor only supports 4 cursor lines: X1, X2, Y1, Y2. In each cursor mode, set or query cursor parameters using the commands in this section.

### :CURSor:IXDelta

#### Query

#### DESCRIPTION

The query returns the current value of cursor 1/(X1-X2).

#### QUERY SYNTAX

:CURSor:IXDelta?

#### RESPONSE FORMAT

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

#### EXAMPLE

Query message:

*CURs:IXD?*

Response message:

*5.7143E+00*

#### RELATED COMMANDS

:CURSor:X1

:CURSor:X2

:CURSor:XDELta

**:CURSor:MITem****Command/Query****DESCRIPTION**

The command specifies the measure item of the cursors, when the cursor mode is measure.

The query returns the current measure item of cursor.

**COMMAND SYNTAX**

**:CURSor:MITem <type>,<source1>[,<source2>]**

**<type>:=** the type of the selected measurement item in advanced measurement,see the table Description of Parameters for details.

**<source1>:=** the source of the selected measurement item in advanced measurement. The optional parameters are the same as the measurement source.

**<source2>:=** when the type is CH Delay type, source2 needs to be specified. The optional parameters are the same as the measurement source

**QUERY SYNTAX**

**:CURSor:MITem?**

**RESPONSE FORMAT**

**<type>,<source1>[,<source2>]**

**EXAMPLE**

The following command sets the mesure item of the cursor to PKPK(C2), when the advanced measurement is turned on.

Command message:

*:CURSor:MITem PKPK,C2*

*CURS:MIT PKPK,C2*

Query message:

*CURS:MIT?*

Response message:

*PKPK,C2*

**:CURSor:MODE****Command/Query****DESCRIPTION**

The command specifies the mode of cursor, and the type of cursor to be displayed when the cursor mode is manual.

The query returns the current mode of cursor.

**COMMAND SYNTAX**

:CURSor:MODE <type>

<type>:= {TRACk|MANual[,<mode>]]MEASure}

<mode>:= {X|Y|XY}

- ◆ MANul means the manual cursors
- ◆ TRACk means the track cursors
- ◆ MEASure means the measure cursors

**QUERY SYNTAX**

:CURSor:MODE?

**RESPONSE FORMAT**

<type>

<type>:= {TRACk|MANual[,<mode>]}

<mode>:= {X|Y|XY}

**EXAMPLE**

The following command sets the cursor type to manual X, when the cursor mode is manual.

Command message:

*:CURSor:MODE MANual,X*

*CURS:MODE MAN,X*

Query message:

*CURS:MODE?*

Response message:

*MANual,X*

**:CURSor:SOURce1****Command/Query****DESCRIPTION**

This command specifies the source of the cursor source 1.

The query returns the current source of the cursor source 1.

**COMMAND SYNTAX**

```
:CURSor:SOURce1 <source>
```

```
<source>:=  
{C<n>|Z<n>|F<x>|M<m>|REF<r>|DIGital|HISTOGRAM}
```

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

```
<r>:= {A|B|C|D}
```

**Note:**

When the cursor mode is a TRACK, the source cannot be set to HISTOGRAM or DIGITAL.

**QUERY SYNTAX**

```
:CURSor:SOURce1?
```

**RESPONSE FORMAT**

```
<source>
```

```
<source>:=  
{C<n>|Z<n>|F<x>|M<m>|REF<r>|DIGital|HISTOGRAM}
```

**EXAMPLE**

The following command sets the source of the cursor source 1 as Channel 1.

Command message:

```
:CURSor:SOURce1 C1  
CURS:SOUR1 C1
```

Query message:

```
CURS:SOUR1?
```

Response message:

```
C1
```

**RELATED COMMANDS**

```
:CURSor:SOURce2
```

**:CURSor:SOURce2****Command/Query****DESCRIPTION**

This command specifies the source of the cursor source 2.

The query returns the current source of the cursor source 2.

**COMMAND SYNTAX**

:CURSor:SOURce2 <source>

<source>:=  
{C<n>|Z<n>|F<x>|M<m>|REF<r>|DIGital|HISTOGRAM}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<r>:= {A|B|C|D}

**Note:**

When the cursor mode is a TRACK, the source cannot be set to HISTOGRAM or DIGital.

**QUERY SYNTAX**

:CURSor:SOURce2?

**RESPONSE FORMAT**

<source>

<source>:=  
{C<n>|Z<n>|F<x>|M<m>|REF<r>|DIGital|HISTOGRAM}

**EXAMPLE**

The following command sets the source of the cursor source 2 as Channel 1.

Command message:

```
:CURSor:SOURce2 C1
CURS:SOUR2 C1
```

Query message:

```
CURS:SOUR2?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:CURSor:SOURce1

**:CURSor:X1****Command/Query****DESCRIPTION**

This command specifies the position of the cursor X1.

The query returns the current position of the cursor X1.

**COMMAND SYNTAX**

:CURSor:X1 <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-horizontal\_grid/2\*timebase+horizontal\_delay, horizontal\_grid/2\*timebase+horizontal\_delay].

**QUERY SYNTAX**

:CURSor:X1?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the value of the cursor X1 to 1 us.

Command message:

```
:CURSor:X1 1.00E-06
```

```
CURS:X1 1.00E-06
```

Query message:

```
CURS:X1?
```

Response message:

```
1.00E-06
```

**RELATED COMMANDS**

:CURSor:X2

:CURSor:XDELta

:CURSor:IXDelta



**:CURSor:X2****Command/Query****DESCRIPTION**

This command specifies the position of the cursor X2.

The query returns the current position of the cursor X2.

**COMMAND SYNTAX**

:CURSor:X2 <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-horizontal\_grid/2\*timebase+horizontal\_delay, horizontal\_grid/2\*timebase+horizontal\_delay].

**QUERY SYNTAX**

:CURSor:X2?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the value of the cursor X2 to 1 us.

Command message:

```
:CURSor:X2 1.00E-06  
CURS:X2 1.00E-06
```

Query message:

```
CURS:X2?
```

Response message:

```
1.00E-06
```

**RELATED COMMANDS**

```
:CURSor:X1  
:CURSor:XDELta  
:CURSor:IXDelta
```

**:CURSor:XDELta****Query****DESCRIPTION**

The query returns the horizontal difference between cursor X1 and cursor X2.

**QUERY SYNTAX**

:CURSor:XDELta?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command returns the current value of the cursor X2-X1.

Query message:

*CURS:XDEL?*

Response message:

*1.750E-01*

**RELATED COMMANDS**

:CURSor:X1

:CURSor:X2

:CURSor:IXDelta

**:CURSor:Y1****Command/Query****DESCRIPTION**

This command specifies the position of the cursor Y1.

The query returns the current position of the cursor Y1.

**COMMAND SYNTAX**

:CURSor:Y1 <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is  $[-\text{vertical\_grid}/2*\text{vertical\_scale}-\text{vertical\_offset}, \text{vertical\_grid}/2*\text{vertical\_scale}-\text{vertical\_offset}]$ .

**QUERY SYNTAX**

:CURSor:Y1?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the value of the cursor Y1 to 12 V.

Command message:

```
:CURSor:Y1 1.20E+01  
CURS:Y1 1.20E+01
```

Query message:

```
CURS:Y1?
```

Response message:

```
1.20E+01
```

**RELATED COMMANDS**

:CURSor:Y2

:CURSor:YDELta

**:CURSor:Y2****Command/Query****DESCRIPTION**

This command specifies the position of the cursor Y2.

The query returns the current position of the cursor Y2.

**COMMAND SYNTAX**

:CURSor:Y2 <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-vertical\_grid/2\*vertical\_scale-vertical\_offset, vertical\_grid/2\*vertical\_scale-vertical\_offset]

**QUERY SYNTAX**

:CURSor:Y2?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the value of the cursor Y2 to 10 V.

Command message:

```
:CURSor:Y2 1.00E+01
```

```
CURS:Y2 1.00E+01
```

Query message:

```
CURS:Y2?
```

Response message:

```
1.00E+01
```

**RELATED COMMANDS**

:CURSor:Y1

:CURSor:YDELta

**:CURSor:YDELta****Query****DESCRIPTION**

The query returns the vertical difference between the cursor Y1 and cursor Y2.

**QUERY SYNTAX**

:CURSor:YDELta?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command returns the current value of the cursor Y2-Y1.

Query message:

*CURS:YDEL?*

Response message:

*1.80E+01*

**RELATED COMMANDS**

:CURSor:Y1

:CURSor:Y2

## Multiple Cursors Commands

The multiple cursors supports 5 cursor modes: Manual X, Manual Y, Track, Measure, and XY. Multiple cursor lines are supported in each mode, and different parameters can be set for each cursor line. Set or query cursor parameters through commands in this section.

**:CURSor:MANual:X<n>**

### Command/Query

#### DESCRIPTION

The command chooses whether to open the specified manual X cursor.

The query returns the current state of the specified manual X cursor.

#### COMMAND SYNTAX

**:CURSor:MANual:X<n> <state>**

**<n>**:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

**<state>**:= {ON|OFF}

#### QUERY SYNTAX

**:CURSor:MANual:X<n>?**

#### RESPONSE FORMAT

**<state>**

**<state>**:= {ON|OFF}

#### EXAMPLE

The following command enables manual X2.

Command message:

```
:CURSor:MANual:X2 ON
CURs:MAN:X2 ON
```

Query message:

```
CURs:MAN:X2?
```

Response message:

```
ON
```

#### RELATED COMMANDS

**:CURSor**

**:CURSor:MANual:X<n>:COLor****Command/Query****DESCRIPTION**

This command sets the color of the specified manual X cursor.

The query returns the color of the specified manual X cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<color>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ DELTA means synchronizing with the color of the reference cursor.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

**QUERY SYNTAX**

:CURSor:MANual:X<n>:COLor?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of hexadecimal RGB color code.

**EXAMPLE**

The following command sets the color of manual X2 to the color of the cursor source.

Command message:

```
:CURSor:MANual:X2:COLor DEFault  
CURS:MAN:X2:COL DEF
```

Query message:

```
CURS:MAN:X2:COL?
```

Response message:

```
"#ffff00"
```

**RELATED COMMANDS**

:CURSor:MANual:X<n>

:CURSor:MANual:X<n>:SOURce

:CURSor:MANual:X<n>:DTCursor

**:CURSor:MANual:X<n>:DFOLlow**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the fixed relative position between the specified manual X cursor and the reference cursor.

The query returns the current state of the fixed relative position between the specified manual X cursor and the reference cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:DFOLlow <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

Only when the reference cursor source is not NONE can it be set.

**QUERY SYNTAX**

:CURSor:MANual:X<n>:DFOLlow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the manual X2 to follow the reference cursor movement status as ON.

Command message:

*:CURSor:MANual:X2:DFOLlow ON  
CURS:MAN:X2:DFOL ON*

Query message:

*CURS:MAN:X2:DFOL?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor:MANual:X<n>:DTCursor



**:CURSor:MANual:X<n>:DTCursor**

**Command/Query**

**DESCRIPTION**

This command sets the reference cursor for the specified manual X cursor.

The query returns the reference cursor for the specified manual X cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:DTCursor <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<source>:= {MX<m>|TRACK<t>|NONE}

- ◆ <m>:= 1 to (# manual X cursor index) in NR1 format.
- ◆ <t>:= 1 to (# track cursor index) in NR1 format
- ◆ NONE means not set reference cursor.

**QUERY SYNTAX**

:CURSor:MANual:X<n>:DTCursor?

**RESPONSE FORMAT**

<source>

<source>:= {MX<m>|TRACK<t>|NONE}

**EXAMPLE**

The following command sets the reference cursor for manual X2 to manual X1.

Command message:

*:CURSor:MANual:X2:DTCursor MX1*  
*CURS:MAN:X2:DTC MX1*

Query message:

*CURS:MAN:X2:DTC?*

Response message:

*MX1*

**RELATED COMMANDS**

:CURSor:MANual:X<n>

:CURSor:TRACK<n>

**:CURSor:MANual:X<n>:DVALue**

**Command/Query**

**DESCRIPTION**

This command sets the position of the specified manual X cursor relative to the reference cursor.

The query returns the position of the specified manual X cursor relative to the reference cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:DVALue <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**QUERY SYNTAX**

:CURSor:MANual:X<n>:DVALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the position of manual relative to the reference cursor to 1us.

Command message:

*:CURSor:MANual:X2:DVALue 1.00E-06*  
*CURS:MAN:X2:DVAL 1.00E-06*

Query message:

*CURS:MAN:X2:DVAL?*

Response message:

*1.00E-06*

**RELATED COMMANDS**

:CURSor:MANual:X<n>:DFOLLOW  
 :CURSor:MANual:X<n>:DTCursor

**:CURSor:MANual:X<n>:LABel****Command/Query****DESCRIPTION**

This command sets the label for the specified manual X cursor.

The query returns the label for the specified manual X cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<state>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ DELTA means synchronizing with the label of the reference cursor.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:CURSor:MANual:X<n>:LABel?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of ASCII text.

**EXAMPLE**

The following command sets the label of manual X2 as the default label.

Command message:

```
:CURSor:MANual:X2:LABel DEFault  
CURS:MAN:X2:LAB DEF
```

Query message:

```
CURS:MAN:X2:LAB?
```

Response message:

```
""
```

**RELATED COMMANDS**

:CURSor:MANual:X<n>

:CURSor:MANual:X<n>:DTCursor

**:CURSor:MANual:X<n>:POSition****Command/Query****DESCRIPTION**

This command sets the position of the specified manual X cursor.

The query to return the position of the specified manual X cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:POSition <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02. The range of the value is [-horizontal\_grid/2\*timebase+horizontal\_delay, horizontal\_grid/2\*timebase+horizontal\_delay].

**QUERY SYNTAX**

:CURSor:MANual:X<n>:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the position of manual X2 to 1us.

Command message:

```
:CURSor:MANual:X2:POSition 1.00E-06
```

```
CURS:MAN:X2:POS 1.00E-06
```

Query message:

```
CURS:MAN:X2:POS?
```

Response message:

```
1.00E-06
```

**RELATED COMMANDS**

:CURSor:MANual:X<n>

**:CURSor:MANual:X<n>:SOURce****Command/Query****DESCRIPTION**

This command sets the source for the specified manual X cursor.

The query to return the source for the specified manual X cursor.

**COMMAND SYNTAX**

:CURSor:MANual:X<n>:SOURce <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the X cursor that is affected by the command.

<source>:=  
{C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|DIGital|ZDIGital  
|HISTOgram }

- ◆ <n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:CURSor:MANual:X<n>:SOURce?

**RESPONSE FORMAT**

<source>

<source>:=  
{C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|DIGital|ZDIGital  
|HISTOgram }

**EXAMPLE**

The following command sets the source of manual X2 to C1.

Command message:

```
:CURSor:MANual:X2:SOURce C1
CURS:MAN:X2:SOUR C1
```

Query message:

```
CURS:MAN:X2:SOUR?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:CURSor:MANual:X<n>

**:CURSor:MANual:Y<n>**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the specified manual Y cursor.

The query returns the current state of the specified manual cursor Y.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n> <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:CURSor:MANual:Y<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables manual Y2.

Command message:

*:CURSor:MANual:Y2 ON*  
*CURs:MAN:Y2 ON*

Query message:

*CURs:MAN:Y2?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor

**:CURSor:MANual:Y<n>:COLor****Command/Query****DESCRIPTION**

This command sets the color of the specified manual Y cursor.

The query returns the color of the specified manual Y cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<color>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ DELTA means synchronizing with the color of the reference cursor.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:COLor?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of hexadecimal RGB color code.

**EXAMPLE**

The following command sets the color of manual Y2 to the color of the cursor source.

Command message:

```
:CURSor:MANual:Y2:COLor DEFault  
CURS:MAN:Y2:COL DEF
```

Query message:

```
CURS:MAN:Y2:COL?
```

Response message:

```
"#ffff00"
```

**RELATED COMMANDS**

:CURSor:MANual:Y<n>

:CURSor:MANual:Y<n>:DTCursor

:CURSor:MANual:Y<n>:SOURce

**:CURSor:MANual:Y<n>:DFOLlow**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the fixed relative position between the specified manual Y cursor and the reference cursor.

The query returns the current state of the fixed relative position between the specified manual Y cursor and the reference cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:DFOLlow <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

Only when the reference cursor source is not NONE can it be set.

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:DFOLlow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the manual Y2 to follow the reference cursor movement status as ON.

Command message:

*:CURSor:MANual:Y2:DFOLlow ON  
CURS:MAN:Y2:DFOL ON*

Query message:

*CURS:MAN:Y2:DFOL?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor:MANual:Y<n>:DTCursor



**:CURSor:MANual:Y<n>:DTCursor****Command/Query****DESCRIPTION**

This command sets the reference cursor for the specified manual Y cursor.

The query returns the reference cursor for the specified manual Y cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:DTCursor <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<source>:= {MY<m>|NONE}

- ◆ <m>:= 1 to (# manual Ycursor index) in NR1 format.
- ◆ NONE means not set reference cursor.

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:DTCursor?

**RESPONSE FORMAT**

<source>

<source>:= {MY<m>|NONE}

**EXAMPLE**

The following command sets the reference cursor for manual Y2 to MY1.

Command message:

```
:CURSor:MANual:Y2:DTCursor MY1  
CURS:MAN:Y2:DTC MY1
```

Query message:

```
CURS:MAN:Y2:DTC?
```

Response message:

```
MY1
```

**RELATED COMMANDS**

:CURSor:MANual:Y<n>

**:CURSor:MANual:Y<n>:DVALue****Command/Query****DESCRIPTION**

This command sets the position of the specified manual Y cursor relative to the reference cursor.

The query returns the position of the specified manual Y cursor relative to the reference cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:DVALue <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:DVALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the position of manual Y2 relative to the reference cursor to 1V.

Command message:

```
:CURSor:MANual:Y2:DVALue 1.00E+00
CURS:MAN:Y2:DVAL 1.00E+00
```

Query message:

```
CURS:MAN:Y2:DVAL?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

```
:CURSor:MANual:Y<n>:DTCursor
:CURSor:MANual:Y<n>:DFOLLOW
```

**:CURSor:MANual:Y<n>:LABel****Command/Query****DESCRIPTION**

This command sets the label for the specified manual Y cursor.

The query returns the label for the specified manual Y cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<state>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ DELTA means synchronizing with the label of the reference cursor.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:LABel?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of ASCII text.

**EXAMPLE**

The following command sets the label of manual Y2 as the default label.

Command message:

```
:CURSor:MANual:Y2:LABel DEFault
CURS:MAN:Y2:LAB DEF
```

Query message:

```
CURS:MAN:Y2:LAB?
```

Response message:

```
""
```

**RELATED COMMANDS**

:CURSor:MANual:Y<n>

:CURSor:MANual:Y<n>:DTCursor

**:CURSor:MANual:Y<n>:POSition****Command/Query****DESCRIPTION**

This command sets the position of the specified manual Y cursor.

The query returns the position of the specified manual Y cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:POSition <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the position of manual Y2 to 1V.

Command message:

*:CURSor:MANual:Y2:POSition 1.00E+00*

*CURS:MAN:Y2:POS 1.00E+00*

Query message:

*CURS:MAN:Y2:POS?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:CURSor:MANual:Y<n>

**:CURSor:MANual:Y<n>:SOURce**

**Command/Query**

**DESCRIPTION**

This command sets the source for the specified manual Y cursor.

The query returns the source for the specified manual Y cursor.

**COMMAND SYNTAX**

:CURSor:MANual:Y<n>:SOURce <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the Y cursor that is affected by the command.

<source>:=

{C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|HISTOGRAM}

- ◆ <n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:CURSor:MANual:Y<n>:SOURce?

**RESPONSE FORMAT**

<source>

<source>:=

{C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|HISTOGRAM}

**EXAMPLE**

The following command sets the source of manual Y2 to C1.

Command message:

*:CURSor:MANual:Y2:SOURce C1*  
*CURS:MAN:Y2:SOUR C1*

Query message:

*CURS:MAN:Y2:SOUR?*

Response message:

*C1*

**RELATED COMMANDS**

:CURSor:MANual:Y<n>

**:CURSor:MEASure<n>****Command/Query****DESCRIPTION**

The command chooses whether to open the specified measure cursor.

This query returns the current state of the specified measure cursor.

**COMMAND SYNTAX**

:CURSor:MEASure<n> <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to MEASure and defines the measure cursor that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:CURSor:MEASure<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables measure cursor MEA2.

Command message:

```
:CURSor:MEASure2 ON  
CURS:MEAS2 ON
```

Query message:

```
CURS:MEAS2?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:CURSor

**:CURSor:MEASure<n>:COLor****Command/Query****DESCRIPTION**

This command sets the color of the specified measure cursor.

The query returns the color of the specified measure cursor.

**COMMAND SYNTAX**

:CURSor:MEASure<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to MEASure and defines the measure cursor that is affected by the command.

<color>:= {DEFault|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

**QUERY SYNTAX**

:CURSor:MEASure<n>:COLor?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of hexadecimal RGB color code.

**EXAMPLE**

The following command sets the color of measure cursor MEA2 to the color of the cursor source.

Command message:

```
:CURSor:MEASure2:COLor DEFault  
CURS:MEAS2:COL DEF
```

Query message:

```
CURS:MEAS2:COL?
```

Response message:

```
"#ffff00"
```

**RELATED COMMANDS**

:CURSor:MEASure<n>

**:CURSor:MEASure<n>:LABel****Command/Query****DESCRIPTION**

This command sets the label for the specified measure cursor MEA.

The query returns the label for the specified measure cursor MEA.

**COMMAND SYNTAX**

:CURSor:MEASure<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to MEASure and defines the measure cursor that is affected by the command.

<state>:= {DEFault|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:CURSor:MEASure<n>:LABel?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of ASCII text.

**EXAMPLE**

The following command sets the label of measure cursor MEA2 as the default label.

Command message:

```
:CURSor:MEASure2:LABel DEFault
CURS:MEAS2:LAB DEF
```

Query message:

```
CURS:MEAS2:LAB?
```

Response message:

```
"MEA2"
```

**RELATED COMMANDS**

:CURSor:MEASure<n>



**:CURSor:MEASure<n>:MITem****Command/Query****DESCRIPTION**

This command sets the measurement items for the specified measurement cursor MEA.

The query to return the measurement items for the specified measurement cursor MEA.

**COMMAND SYNTAX**

:CURSor:MEASure<n>:MITem <type>,<src1>[,<src2>]

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to MEASure and defines the measure cursor that is affected by the command.

<type>:= The parameters are the same as the measurement type, see Table "Description of Parameters" for details.

<source1>:= the source of the selected measurement item in advanced measurement. The optional parameters are the same as the measurement source.

<source2>:= when the type is CH Delay type, source2 needs to be specified. The optional parameters are the same as the measurement source

**QUERY SYNTAX**

:CURSor:MEASure<n>:MITem?

**RESPONSE FORMAT**

<type>,<source1>[,<source2>]

**EXAMPLE**

When advanced measurement is turned on, the following command sets the measurement item of the measurement cursor MEA2 to PKPK (C2).

Command message:

```
:CURSor:MEASure2:MITem PKPK,C2
CURS:MEAS2:MIT PKPK,C2
```

Query message:

```
CURS:MEAS2:MIT?
```

Response message:

```
PKPK,C2
```

**RELATED COMMANDS**

```
:CURSor:MEASure<n>
:MEASure:ADVanced:P<n>:TYPE
```

**:CURSor:TRACk<n>**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the specified track cursor TX.

The query returns the current state of the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n> <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:CURSor:TRACk<n>?

**RESPONSE FORMAT**

{ON|OFF}

**EXAMPLE**

The following command enables track cursor TX2.

Command message:  
*:CURSor:TRACk2 ON*  
*CURs:TRAC2 ON*

Query message:  
*CURs:TRAC2?*

Response message:  
*ON*

**RELATED COMMANDS**

:CURSor

**:CURSor:TRACk<n>:COLor****Command/Query****DESCRIPTION**

This command sets the color of the specified track cursor TX.

The query returns the color of the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<color>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ DELTA means synchronizing with the color of the reference cursor.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

**QUERY SYNTAX**

:CURSor:TRACk<n>:COLor?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of hexadecimal RGB color code.

**EXAMPLE**

The following command sets the color of track cursor TX2 to the color of the cursor source.

Command message:

```
:CURSor:TRACk2:COLor DEFault  
CURS:TRACk2:COL DEF
```

Query message:

```
CURS:TRACk2:COL?
```

Response message:

```
"#ffff00"
```

**RELATED COMMANDS**

:CURSor:TRACk<n>

:CURSor:TRACk<n>:DTCursor

:CURSor:TRACk<n>:SOURce

**:CURSor:TRACk<n>:DFOLlow****Command/Query****DESCRIPTION**

The command chooses whether to open the fixed relative position between the specified track cursor TX and the reference cursor.

The query returns the current state of the fixed relative position between the specified track cursor TX and the reference cursor.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:DFOLlow <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

Only when the reference cursor source is not NONE can it be set.

**QUERY SYNTAX**

:CURSor:TRACk<n>:DFOLlow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the track cursor TX2 to follow the reference cursor movement status as ON.

Command message:

```
:CURSor:TRACk2:DFOLlow ON
CURS:TRAC2:DFOL ON
```

Query message:

```
CURS:MAN:X2:DFOL?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:CURSor:TRACk<n>:DTCursor

**:CURSor:TRACk<n>:DTCursor****Command/Query****DESCRIPTION**

This command sets the reference cursor for the specified track cursor TX.

The query returns the reference cursor for the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:DTCursor <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<source>:= {MX<m>|TRACk<t>|NONE}

- ◆ <m>:= 1 to (# manual X cursor index) in NR1 format.
- ◆ <t>:= 1 to (# track cursor index) in NR1 format
- ◆ NONE means not set reference cursor.

**QUERY SYNTAX**

:CURSor:TRACk<n>:DTCursor?

**RESPONSE FORMAT**

<source>

<source>:= {MX<m>|TRACk<t>|NONE}

**EXAMPLE**

The following command sets the reference cursor for track cursor TX2 to MX1.

Command message:

```
:CURSor:TRACk:DTCursor MX1  
CURS:TRAC2:DTC MX1
```

Query message:

```
CURS:TRAC2:DTC?
```

Response message:

```
MX1
```

**RELATED COMMANDS**

:CURSor:MANual:X<n>

:CURSor:TRACk<n>

**:CURSor:TRACk<n>:DVALue**

**Command/Query**

**DESCRIPTION**

This command sets the position of the specified track cursor TX relative to the reference cursor.

The query returns the position of the specified track cursor TX relative to the reference cursor.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:DVALue <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**QUERY SYNTAX**

:CURSor:TRACk<n>:DVALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the position of track cursor TX2 relative to the reference cursor to 1us.

Command message:

*:CURSor:TRACk2:DVALue 1.00E-06*  
*CURS:TRAC2:DVAL 1.00E-06*

Query message:

*CURS:TRAC2:DVAL?*

Response message:

*1.00E-06*

**RELATED COMMANDS**

:CURSor:TRACk<n>:DTCursor  
 :CURSor:TRACk<n>:DFOLlow

**:CURSor:TRACk<n>:LABel****Command/Query****DESCRIPTION**

This command sets the label for the specified track cursor TX.

The query returns the label for the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<state>:= {DEFault|DELTa|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ DELTa means synchronizing with the label of the reference cursor.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:CURSor:TRACk<n>:LABel?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of ASCII text.

**EXAMPLE**

The following command sets the label of track cursor TX2 as the default label.

Command message:

```
:CURSor:TRACk2:LABel DEFault  
CURS:TRAC2:LAB DEF
```

Query message:

```
CURS:TRAC2:LAB?
```

Response message:

```
""
```

**RELATED COMMANDS**

:CURSor:TRACk<n>

:CURSor:TRACk<n>:DTCursor

**:CURSor:TRACk<n>:POSition****Command/Query****DESCRIPTION**

This command sets the horizontal position of the specified track cursor TX.

The query returns the horizontal position of the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:POSition <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02, The range of the value is [-horizontal\_grid/2\*timebase+horizontal\_delay, horizontal\_grid/2\*timebase+horizontal\_delay].

**QUERY SYNTAX**

:CURSor:TRACk<n>:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command sets the horizontal position of track cursor TX2 to 1us.

Command message:

```
:CURSor:TRACk2:POSition 1.00E-06
CURS:TRAC2:POS 1.00E-06
```

Query message:

```
CURS:TRAC2:POS?
```

Response message:

```
1.00E-06
```

**RELATED COMMANDS**

:CURSor:TRACk<n>



**:CURSor:TRACk<n>:SOURce****Command/Query****DESCRIPTION**

This command sets the source for the specified track cursor TX.

The query returns the source for the specified track cursor TX.

**COMMAND SYNTAX**

:CURSor:TRACk<n>:SOURce <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACk and defines the track cursor that is affected by the command.

<source>:= {C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>}

- ◆ <n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.
- ◆ <m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:CURSor:TRACk<n>:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>}

**EXAMPLE**

The following command sets the source of track cursor TX2 to C1.

Command message:

```
:CURSor:TRACk2:SOURce C1
CURS:TRAC2:SOUR C1
```

Query message:

```
CURS:TRAC2:SOUR?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:CURSor:TRACk<n>

**:CURSor:TRACk<n>:VALue****Query****DESCRIPTION**

The query returns the vertical position of the specified track cursor TX.

**QUERY SYNTAX**

:CURSor:TRACk<n>:VALue?

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to TRACK and defines the track cursor that is affected by the command.

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E-02

**EXAMPLE**

The following command query returns the vertical position of track cursor TX2.

Query message:

*CURs:TRAC2:VAL?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:CURSor:TRACk<n>:POSition

**:CURSor:XY:X<n>**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the specified XY cursor X.

The query returns the current state of the specified XY cursor X.

**COMMAND SYNTAX**

:CURSor:XY:X<n> <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

- ◆ The XY cursor can only be turned on in XY mode.
- ◆ When XY mode is turned off, the XY cursor automatically closes

**QUERY SYNTAX**

:CURSor:XY:X<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables XY cursor X2.

Command message:

*:CURSor:XY:X2 ON*  
*CURS:XY:X2 ON*

Query message:

*CURS:XY:X2?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor  
 :ACQuire:MODE

**:CURSor:XY:X<n>:COLor**

### Command/Query

#### DESCRIPTION

This command sets the color of the specified XY cursor X.

The query returns the color of the specified XY cursor X.

#### COMMAND SYNTAX

:CURSor:XY:X<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<color>:= {DEFault|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

#### QUERY SYNTAX

:CURSor:XY:X<n>:COLor?

#### RESPONSE FORMAT

<string>

<string>:= Quoted string of hexadecimal RGB color code.

#### EXAMPLE

The following command sets the color of XY cursor X2 to the color of the cursor source.

Command message:

```
:CURSor:XY:X2:COLor DEFault
CURS:XY:X2:COL DEF
```

Query message:

```
CURS:XY:X2:COL?
```

Response message:

```
"#ffff00"
```

#### RELATED COMMANDS

:CURSor:XY:X<n>

:CURSor:XY:X<n>:DTCursor

**:CURSor:XY:X<n>:DFOLlow****Command/Query****DESCRIPTION**

The command chooses whether to open the fixed relative position between the specified XY cursor X and the reference cursor.

The query returns the current state of the fixed relative position between the specified XY cursor X and the reference cursor.

**COMMAND SYNTAX**

:CURSor:XY:X<n>:DFOLlow <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

Only when the reference cursor source is not NONE can it be set.

**QUERY SYNTAX**

:CURSor:XY:X<n>:DFOLlow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the XY cursor X2 to follow the reference cursor movement status as ON.

Command message:

```
:CURSor:XY:X2:DFOLlow ON  
CURs:XY:X2:DFOL ON
```

Query message:

```
CURs:XY:X2:DFOL?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:CURSor:XY:X<n>:DTCursor

**:CURSor:XY:X<n>:DTCursor****Command/Query****DESCRIPTION**

This command sets the reference cursor for the specified XY cursor X.

The query returns the reference cursor for the specified XY cursor X.

**COMMAND SYNTAX**

:CURSor:XY:X<n>:DTCursor <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<source>:= {XY\_X<m>|NONE}

- ◆ <m>:= 1 to (#XY X cursor index) in NR1 format.
- ◆ NONE means not set reference cursor.

**QUERY SYNTAX**

:CURSor:XY:X<n>:DTCursor?

**RESPONSE FORMAT**

<source>

<source>:= { XY\_X<m>|NONE}

**EXAMPLE**

The following command sets the reference cursor for XY cursor X2 to XY\_X1.

Command message:

```
:CURSor:XY:X2:DTCursor XY_X1
CURS:XY:X2:DTC XY_X1
```

Query message:

```
CURS:XY:X2:DTC?
```

Response message:

```
XY_X1
```

**RELATED COMMANDS**

:CURSor:XY:X<n>

**:CURSor:XY:X<n>:DVALue****Command/Query****DESCRIPTION**

This command sets the position of the specified XY cursor X relative to the reference cursor.

The query returns the position of the specified XY cursor X relative to the reference cursor.

**COMMAND SYNTAX**

:CURSor:XY:X<n>:DVALue <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**QUERY SYNTAX**

:CURSor:XY:X<n>:DVALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**EXAMPLE**

The following command sets the position of XY X1 relative to the reference cursor to 1V.

Command message:

```
:CURSor:XY:X2:DVALue 1.00E+00  
CURs:XY:X2:DVAL 1.00E+00
```

Query message:

```
CURs:XY:X2:DVAL?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:CURSor:XY:X<n>:DFOLlow  
:CURSor:XY:X<n>:DTCursor

**:CURSor:XY:X<n>:LABel**

### Command/Query

#### DESCRIPTION

This command sets the label for the specified XY cursor X.

The query returns the label for the specified XY cursor X.

#### COMMAND SYNTAX

:CURSor:XY:X<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<state>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ DELTA means synchronizing with the label of the reference cursor.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

#### QUERY SYNTAX

:CURSor:XY:X<n>:LABel?

#### RESPONSE FORMAT

<string>

<string>:= Quoted string of ASCII text.

#### EXAMPLE

The following command sets the label of XY cursor X2 as the default label.

Command message:

```
:CURSor:XY:X2:LABel DEFault
CURS:XY:X2:LAB DEF
```

Query message:

```
CURS:XY:X2:LAB?
```

Response message:

```
"XY"
```

#### RELATED COMMANDS

:CURSor:XY:X<n>

:CURSor:XY:X<n>:DTCursor



**:CURSor:XY:X<n>:POSition****Command/Query****DESCRIPTION**

This command sets the position of the specified XY cursor X.

The query returns the position of the specified XY cursor X.

**COMMAND SYNTAX**

:CURSor:XY:X<n>:POSition <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to X and defines the XY X cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00.

**QUERY SYNTAX**

:CURSor:XY:X<n>:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**EXAMPLE**

The following command sets the position of XY cursor X2 to 1V.

Command message:

```
:CURSor:XY:X2:POSition 1.00E+00  
CURS:XY:X2:POS 1.00E+00
```

Query message:

```
CURS:XY:X2:POS?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:CURSor:XY:X<n>

**:CURSor:XY:Y<n>**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the specified XY cursor Y.

The query returns the current state of the specified XY cursor Y.

**COMMAND SYNTAX**

:CURSor:XY:Y<n> <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

- ◆ The XY cursor can only be turned on in XY mode.
- ◆ When XY mode is turned off, the XY cursor automatically closes

**QUERY SYNTAX**

:CURSor:XY:Y<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables XY cursor Y2.

Command message:

*:CURSor:XY:Y2 ON*

*CURS:XY:Y2 ON*

Query message:

*CURS:XY:Y2?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor

:ACQuire:MODE

**:CURSor:XY:Y<n>:COLor**

**Command/Query**

**DESCRIPTION**

This command sets the color of the specified XY cursor Y.

The query returns the color of the specified XY cursor Y.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:COLor <color>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<color>:= {DEFault|CUSTom[,<string>]}

- ◆ DEFault means color of cursor source.
- ◆ CUSTom means set to the <string>.

<string>:= Quoted string of hexadecimal RGB color code.

**QUERY SYNTAX**

:CURSor:XY:Y<n>:COLor?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of hexadecimal RGB color code.

**EXAMPLE**

The following command sets the color of XY cursor Y2 to the color of the cursor source.

Command message:

*:CURSor:XY:Y2:COLor DEFault*  
*CURS:XY:Y2:COL DEF*

Query message:

*CURS:XY:Y2:COL?*

Response message:

*"#ffff00"*

**RELATED COMMANDS**

:CURSor:XY:Y<n>

:CURSor:XY:Y<n>:DTCursor

**:CURSor:XY:Y<n>:DFOLlow**

**Command/Query**

**DESCRIPTION**

The command chooses whether to open the fixed relative position between the specified XY cursor Y and the reference cursor.

The query returns the current state of the fixed relative position between the specified XY cursor Y and the reference cursor.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:DFOLlow <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<state>:= {ON|OFF}

**Note:**

Only when the reference cursor source is not NONE can it be set.

**QUERY SYNTAX**

:CURSor:XY:Y<n>:DFOLlow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the XY cursor Y2 to follow the reference cursor movement status as ON.

Command message:

*:CURSor:XY:Y2:DFOLlow ON*  
*CURS:XY:Y2:DFOL ON*

Query message:

*CURS:XY:Y2:DFOL?*

Response message:

*ON*

**RELATED COMMANDS**

:CURSor:XY:Y<n>:DTCursor

**:CURSor:XY:Y<n>:DTCursor****Command/Query****DESCRIPTION**

This command sets the reference cursor for the specified XY cursor Y.

The query returns the reference cursor for the specified XY cursor Y.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:DTCursor <source>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<source>:= {XY\_Y<m>|NONE}

- ◆ <m>:= 1 to (#XY Y cursor index) in NR1 format.
- ◆ NONE means not set reference cursor.

**QUERY SYNTAX**

:CURSor:XY:Y<n>:DTCursor?

**RESPONSE FORMAT**

<source>

<source>:= { XY\_Y<m>|NONE}

**EXAMPLE**

The following command sets the reference cursor for XY cursor Y2 to XY\_Y1.

Command message:

```
:CURSor:XY:Y2:DTCursor XY_Y1  
CURS:XY:Y2:DTC XY_Y1
```

Query message:

```
CURS:XY:Y2:DTC?
```

Response message:

```
XY_Y1
```

**RELATED COMMANDS**

:CURSor:XY:Y<n>

**:CURSor:XY:Y<n>:DVALue****Command/Query****DESCRIPTION**

This command sets the position of the specified XY cursor Y relative to the reference cursor.

The query returns the position of the specified XY cursor Y relative to the reference cursor.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:DVALue <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**QUERY SYNTAX**

:CURSor:XY:Y<n>:DVALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**EXAMPLE**

The following command sets the position of XY cursor Y2 relative to the reference cursor to 1V.

Command message:

```
:CURSor:XY:Y2:DVALue 1.00E+00
CURs:XY:Y2:DVAL 1.00E+00
```

Query message:

```
CURs:XY:Y2:DVAL?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

```
:CURSor:XY:Y<n>:DFOLlow
:CURSor:XY:Y<n>:DTCursor
```

**:CURSor:XY:Y<n>:LABel****Command/Query****DESCRIPTION**

This command sets the label for the specified XY cursor Y.

The query returns the label for the specified XY cursor Y.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:LABel <state>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<state>:= {DEFault|DELTA|CUSTom[,<string>]}

- ◆ DEFault means that the label is empty.
- ◆ DELTA means synchronizing with the label of the reference cursor.
- ◆ CUSTom means set to <string>.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:CURSor:XY:Y<n>:LABel?

**RESPONSE FORMAT**

<string>

<string>:= Quoted string of ASCII text.

**EXAMPLE**

The following command sets the label of XY cursor Y2 as the default label.

Command message:

*:CURSor:XY:Y2:LABel DEFault*

*CURS:XY:Y2:LAB DEF*

Query message:

*CURS:XY:Y2:LAB?*

Response message:

*"XY"*

**RELATED COMMANDS**

:CURSor:XY:Y<n>

:CURSor:XY:Y<n>:DTCursor

**:CURSor:XY:Y<n>:POSition****Command/Query****DESCRIPTION**

This command sets the position of the specified XY cursor Y.

The query returns the position of the specified XY cursor Y.

**COMMAND SYNTAX**

:CURSor:XY:Y<n>:POSition <value>

<n>:= 1 to (# cursor index) in NR1 format, is attached as a suffix to Y and defines the XY Y cursor that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00.

**QUERY SYNTAX**

:CURSor:XY:Y<n>:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+00

**EXAMPLE**

The following command sets the position of XY cursor Y2 to 1V.

Command message:

```
:CURSor:XY:Y2:POSition 1.00E+00
CURS:XY:Y2:POS 1.00E+00
```

Query message:

```
CURS:XY:Y2:POS?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:CURSor:XY:Y<n>



## DECode Commands

The :DECode subsystem commands control the basic decode functions of the oscilloscope.

- ◆ **:DECode**
- ◆ **:DECode:LIST**
- ◆ **:DECode:LIST:LINE**
- ◆ **:DECode:LIST:SCROLL**
- ◆ **:DECode:BUS<n>**
- ◆ **:DECode:BUS<n>:COPY**
- ◆ **:DECode:BUS<n>:FORMat**
- ◆ **:DECode:BUS<n>:PROTOcol**
- ◆ **:DECode:BUS<n>:RESult**
- ◆ **:DECode:BUS<n>:IIC Commands**
- ◆ **:DECode:BUS<n>:SPI Commandds**
- ◆ **:DECode:BUS<n>:UART Commands**
- ◆ **:DECode:BUS<n>:CAN Commands**
- ◆ **:DECode:BUS<n>:LIN Commands**
- ◆ **:DECode:BUS<n>:CANFd Commands [Option]**
- ◆ **:DECode:BUS<n>:IIS Commands [Option]**
- ◆ **:DECode:BUS<n>:M1553 Commands [Option]**
- ◆ **:DECode:BUS<n>:SENT Commands [Option]**
- ◆ **:DECode:BUS<n>:MANChester Commands [Option]**

**:DECode****Command/Query****DESCRIPTION**

The command sets the state of the decode function.

This query returns the current status of the decode function.

**COMMAND SYNTAX**

:DECode <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DECode?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the decode function.

Command message:

*:DECode ON*

*DEC ON*

Query message:

*DEC?*

Response message:

*ON*

**:DECode:LIST****Command/Query****DESCRIPTION**

The command enables or disables the list of decode result.

This query returns the current switch state of the decode list.

**COMMAND SYNTAX**

:DECode:LIST <state>

<state>:= {OFF|D1|D2}

- ◆ D1 means bus 1
- ◆ D2 means bus 2

**QUERY SYNTAX**

:DECode:LIST?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|D1|D2}

**EXAMPLE**

The following command enables the D1 list.

Command message:

*:DECode:LIST D1*  
*DEC:LIST D1*

Query message:

*DEC:LIST?*

Response message:

*D1*

**:DECode:LIST:LINE****Command/Query****DESCRIPTION**

The command sets the number of lines displayed in the decoding list on the screen.

This query returns the number of lines displayed in the decoding list.

**COMMAND SYNTAX**

:DECode:LIST:LINE <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of value is [1, 7].

**QUERY SYNTAX**

:DECode:LIST:LINE?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of lines displayed by decoding to 6.

Command message:

*:DECode:LIST:LINE 6*

*DEC:LIST:LINE 6*

Query message:

*DEC:LIST:LINE?*

Response message:

*6*

**:DECode:LIST:SCRoll****Command/Query****DESCRIPTION**

The command sets the selected line when the decode list is turned on.

This query returns the selected line of the decode list.

**COMMAND SYNTAX**

:DECode:LIST:SCRoll <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:LIST:SCRoll?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the third line 3 selected when decoding the display.

Command message:

*:DECode:LIST:SCRoll 3*

*DEC:LIST:SCR 3*

Query message:

*DEC:LIST:SCR?*

Response message:

*3*

**RELATED COMMANDS**

:DECode:LIST

:DECode:LIST:LINE

**:DECode:LIST<n>:RESult****Query****DESCRIPTION**

This query returns the result of the specified decoding list.

**QUERY SYNTAX**

:DECode:LIST<n>:RESult?

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

**RESPONSE FORMAT**

<ascii\_text>

The data is separated by commas, and each frame is separated by LF(\n). The data value is related to the format set by “:DECode:BUS<n>:FORMat”. The first row of data is header description information.

**EXAMPLE**

The following command sets the decoding protocol of bus 1 to IIC.

Query message:

*DEC:BUS1:RES?*

Response message:

*UART,\sTime,\sRx,\sRx\sErr,\sTx,\sTx\sErr\s*

*1,-93.7501us,0x4C,\s,\s,\s*

*2,-93.7501us,\s,\s,0x4C,\s*

*3,-4.20950us,0x45,\s,\s,\s*

*4,-4.20950us,\s,\s,0x45,\s*

*5,85.4178us,,Less\sLength,\s,\s*

*6,85.4178us,\s,\s,,Less\sLength*

**:DECode:BUS<n>****Command/Query****DESCRIPTION**

The command sets the status of the decode bus.

This query returns the current status of the decode bus.

**COMMAND SYNTAX**

:DECode:BUS<n> <state>

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

<state>:= {ON|OFF}.

**QUERY SYNTAX**

:DECode:BUS<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets decode bus 1 on.

Command message:

*:DECode:BUS1 ON*

*DEC:BUS1 ON*

Query message:

*DEC:BUS1?*

Response message:

*ON*

**RELATED COMMANDS**

:DECode

**:DECode:BUS<n>:COPY****Command****DESCRIPTION**

The command synchronizes the decoding settings with the trigger settings.

**COMMAND SYNTAX**

:DECode:BUS<n>:COPY <operation>

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

<operation>:= {FROMtrigger|TOTRigger}.

- ◆ FROMtrigger means copy trigger settings to the decoding bus.
- ◆ TOTRigger means copy decoding settings to trigger.

**EXAMPLE**

The following command copies the decode settings on bus 1 to the trigger settings.

Command message:

```
:DECode:BUS1:COPY FROMtrigger  
DEC:BUS1:COPY FROM
```



**:DECode:BUS<n>:FORMat****Command/Query****DESCRIPTION**

The command selects the display format of the specified decode bus.

This query returns the display format of the specified decode bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:FORMat <format>

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

<format>:= {BINary|DECimal|HEX|ASCii}

**QUERY SYNTAX**

:DECode:BUS<n>:FORMat?

**RESPONSE FORMAT**

<format>

<format>:= {BINary|DECimal|HEX|ASCii}

**EXAMPLE**

The following command selects the display format of the bus 1 as HEX.

Command message:

*:DECode:BUS1:FORMat HEX*

*DEC:BUS1:FORM HEX*

Query message:

*DEC:BUS1:FORM?*

Response message:

*HEX*

**:DECode:BUS<n>:PROTocol**

### Command/Query

#### DESCRIPTION

The command selects the protocol of the specified bus.

This query returns the protocol of the specified bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:PROTocol <protocol>

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

<protocol>:=  
{IIC|SPI|UART|CAN|LIN|FLEXray|CANFd|IIS|M1553|SENT|Manchester|A429|USB20}

#### QUERY SYNTAX

:DECode:BUS<n>:PROTocol?

#### RESPONSE FORMAT

<protocol>

<protocol>:=  
{IIC|SPI|UART|CAN|LIN|FLEXray|CANFd|IIS|M1553}

#### EXAMPLE

The following command sets the decoding protocol of bus 1 to IIC.

Command message:

```
:DECode:BUS1:PROTocol IIC  
DEC:BUS1:PROT IIC
```

Query message:

```
DEC:BUS1:PROT?
```

Response message:

```
IIC
```

**:DECode:BUS<n>:RESult****Query****DESCRIPTION**

This query returns the result of the specified decoding bus. When there is a large amount of decoded data on the screen, the display space of the decoding bus is limited, and the returned results will also be incomplete. If you need to obtain full decoded data, use the command :DECode:LIST<n>:RESult.

**QUERY SYNTAX**

:DECode:BUS<n>:RESult?

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

**RESPONSE FORMAT**

<ascii\_text>

The data is separated by commas, and each frame is separated by semicolons and automatically wrapped. The data value is related to the format set by “:DECode:BUS<n>:FORMat”. The first row of data is header description information.

**EXAMPLE**

The following command sets the decoding protocol of bus 1 to IIC.

Query message:

*DEC:BUS1:RES?*

Response message:

*lin, sync, id, parity, data, checksum;  
0x55, 0x06, 0, 0x54\|s0x5F, 0x46;*

**:DECode:BUS<n>:IIC Commands**

The :DECode:BUS<n>:IIC subsystem commands control the IIC decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:IIC:RWBit**
- ◆ **:DECode:BUS<n>:IIC:SCLSource**
- ◆ **:DECode:BUS<n>:IIC:SCLThreshold**
- ◆ **:DECode:BUS<n>:IIC:SDASource**
- ◆ **:DECode:BUS<n>:IIC:SDAThreshold**

**:DECode:BUS<n>:IIC:RWBit****Command/Query****DESCRIPTION**

This command selects whether the decoding result includes read bit and write bit.

This query returns whether the decoding result includes read and write bits.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIC:RWBit <state>

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

<state>:= {ON|OFF}.

**QUERY SYNTAX**

:DECode:BUS<n>:IIC:RWBit?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command selects to enable read and write bits on bus 1.

Command message:

*:DECode:BUS1:IIC:RWBit ON*  
*DEC:BUS1:IIC:RWB ON*

Query message:

*DEC:BUS1:IIC:RWB?*

Response message:

*ON*

**:DECode:BUS<n>:IIC:SCLSource****Command/Query****DESCRIPTION**

The command selects the SCL source of the IIC bus.

This query returns the current SCL source of the IIC bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIC:SCLSource <source>

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

<source>:= {C<n>|D<d>}.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:IIC:SCLSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the SCL source of the IIC on bus 1 as C1.

Command message:

```
:DECode:BUS1:IIC:SCLSource C1
DEC:BUS1:IIC:SCLS C1
```

Query message:

```
DEC:BUS1:IIC:SCLS?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:DECode:BUS<n>:IIC:SCLThreshold

:DECode:BUS<n>:IIC:SDASource

**:DECode:BUS<n>:IIC:SCLThreshold****Command/Query****DESCRIPTION**

The command sets the threshold of the SCL on IIC bus.

This query returns the current threshold of the SCL on IIC bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIC:SCLThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:IIC:SCLThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the SCL to 1 V on bus 1.

Command message:

```
:DECode:BUS1:IIC:SCLThreshold 1.00E+00  
DEC:BUS1:IIC:SCLT 1.00E+00
```

Query message:

```
DEC:BUS1:IIC:SCLT?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:DECode:BUS<n>:IIC:SCLSource

**:DECode:BUS<n>:IIC:SDASource****Command/Query****DESCRIPTION**

The command selects the SDA source of the IIC bus.

This query returns the current SDA source of the IIC bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIC:SDASource <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:IIC:SDASource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the SDA source of the IIC on bus 1 as C1.

Command message:

*:DECode:BUS1:IIC:SDASource C1*

*DEC:BUS1:IIC:SDAS C1*

Query message:

*DEC:BUS1:IIC:SDAS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:IIC:SDAThreshold

:DECode:BUS<n>:IIC:SCLSource



**:DECode:BUS<n>:IIC:SDAThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the SDA on IIC bus.

This query returns the current threshold of the SDA on IIC bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIC:SDAThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:IIC:SDAThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the SDA to 1 V on bus 1.

Command message:

*:DECode:BUS1:IIC:SDAThreshold 1.00E+00  
DEC:BUS1:IIC:SDAT 1.00E+00*

Query message:

*DEC:BUS1:IIC:SDAT?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:IIC:SDASource

**:DECode:BUS<n>:SPI Commandds**

The :DECode:BUS<n>:SPI subsystem commands control the SPI decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:SPI:BITOrder**
- ◆ **:DECode:BUS<n>:SPI:CLKSource**
- ◆ **:DECode:BUS<n>:SPI:CLKThreshold**
- ◆ **:DECode:BUS<n>:SPI:CSSource**
- ◆ **:DECode:BUS<n>:SPI:CSThreshold**
- ◆ **:DECode:BUS<n>:SPI:CSType**
- ◆ **:DECode:BUS<n>:SPI:DLEnGth**
- ◆ **:DECode:BUS<n>:SPI:LATChedge**
- ◆ **:DECode:BUS<n>:SPI:MISOSource**
- ◆ **:DECode:BUS<n>:SPI:MISOThreshold**
- ◆ **:DECode:BUS<n>:SPI:MOSISource**
- ◆ **:DECode:BUS<n>:SPI:MOSIThreshold**
- ◆ **:DECode:BUS<n>:SPI:NCSSource**
- ◆ **:DECode:BUS<n>:SPI:NCSThreshold**

**:DECode:BUS<n>:SPI:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order of the SPI bus.

This query returns the current bit order of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:BITorder <order>

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

<order>:= {LSB|MSB}.

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:BITorder?

**RESPONSE FORMAT**

<order>

<order>:= {LSB|MSB}

**EXAMPLE**

The following command sets bit order of the SPI on BUS 1 to LSB.

Command message:

*:DECode:BUS1:SPI:BITorder LSB*

*DEC:BUS1:SPI:BIT LSB*

Query message:

*DEC:BUS1:SPI:BIT?*

Response message:

*LSB*

**:DECode:BUS<n>:SPI:CLKSource****Command/Query****DESCRIPTION**

The command selects the CLK source of the SPI bus.

This query returns the current CLK source of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:CLKSource <source>

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

<source>:= {C<n>|D<d>}.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:CLKSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the CLK source of the SPI on bus 1 as C1.

Command message:

*:DECode:BUS1:SPI:CLKSource C1*

*DEC:BUS1:SPI:CLKS C1*

Query message:

*DEC:BUS1:SPI:CLKS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:CLKThreshold

**:DECode:BUS<n>:SPI:CLKThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the CLK on SPI bus.

This query returns the current threshold of the CLK on SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:CLKThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:CLKThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the CLK to 1 V on bus 1.

Command message:  
*:DECode:BUS1:SPI:CLKThreshold 1.00E+00*  
*DEC:BUS1:SPI:CLKT 1.00E+00*  
 Query message:  
*DEC:BUS1:SPI:CLKT?*  
 Response message:  
*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:CLKSource

**:DECode:BUS<n>:SPI:CSSource**

### Command/Query

#### DESCRIPTION

The command sets the CS source of the SPI bus.

This query returns the current CS source of the SPI bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:SPI:CSSource <source>

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

<source>:= {C<n>|D<d>}.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:SPI:CSSource?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command sets the CS source of the SPI on bus 1 as C1.

Command message:

*:DECode:BUS1:SPI:CSSource C1*

*DEC:BUS1:SPI:CSS C1*

Query message:

*DEC:BUS1:SPI:CSS?*

Response message:

*C1*

**:DECode:BUS<n>:SPI:CSThreshold****Command/Query****DESCRIPTION**

The command sets the threshold of the CS on SPI bus.

This query returns the current threshold of the CS on SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:CSThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:CSThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the CS to 1 V on bus 1.

Command message:

*:DECode:BUS1:SPI:CSThreshold 1.00E+00*

*DEC:BUS1:SPI:CST 1.00E+00*

Query message:

*DEC:BUS1:SPI:CST?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:CLKSource

**:DECode:BUS<n>:SPI:CSType****Command/Query****DESCRIPTION**

The command sets the chip selection type of the SPI bus.

This query returns the current chip selection type of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:CSType <type>

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

<type>:= {NCS|CS|TIMEout[,<time>]}

- ◆ CS means set to chip select state.
- ◆ NCS means set to non-chip select state.
- ◆ TIMEout indicates set to clock timeout status.

<time>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [1.00E-07, 5.00E-03].

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:CSType?

**RESPONSE FORMAT**

<type>

<type>:= {NCS|CS|TIMEout[,<time>]}

<time>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the chip selection type of the SPI on bus 1 to CS.

Command message:

```
:DECode:BUS1:SPI:CSType CS
DEC:BUS1:SPI:CSTY CS
```

Query message:

```
DEC:BUS1:SPI:CSTY?
```

Response message:

```
CS
```



**:DECode:BUS<n>:SPI:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the SPI bus.

This query returns the current data length of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:DLENgth <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [4, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:DLENgth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the SPI on bus 1 to 5.

Command message:

*:DECode:BUS1:SPI:DLENgth 5*  
*DEC:BUS1:SPI:DLEN 5*

Query message:

*DEC:BUS1:SPI:DLEN?*

Response message:

*5*

**:DECode:BUS<n>:SPI:LATChedge****Command/Query****DESCRIPTION**

The command selects the sampling edge of CLK on SPI bus.

This query returns the sampling edge of CLK on SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:LATChedge <slope>

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

<slope>:= {RISing|FALLing}

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:LATChedge?

**RESPONSE FORMAT**

<slope>

<slope>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the threshold judgment condition of CLK on bus 1 to RISing.

Command message:

*:DECode:BUS1:SPI:LATChedge RISing*  
*DEC:BUS1:SPI:LATC RIS*

Query message:

*DEC:BUS1:SPI:LATC?*

Response message:

*RISing*

**:DECode:BUS<n>:SPI:MISOSource****Command/Query****DESCRIPTION**

The command selects the MISO source of the SPI bus.

This query returns the current MISO source of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:MISOSource <source>

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

<source>:= {C<n>|D<d>|DIS}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1. For example, C1 selects analog channel 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1. For example, D1 selects digital channel 1.

- ◆ DIS means no source selected.

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:MISOSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>|DIS}

**EXAMPLE**

The following command sets the MISO source of the SPI on bus 1 as C1.

Command message:

*:DECode:BUS1:SPI:MISOSource C1*

*DEC:BUS1:SPI:MISOS C1*

Query message:

*DEC:BUS1:SPI:MISOS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:MISOThreshold

**:DECode:BUS<n>:SPI:MISOThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the MISO on SPI bus.

This query returns the current threshold of the MISO.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:MISOThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:MISOThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the MISO to 1 V on bus 1.

Command message:

*:DECode:BUS1:SPI:MISOThreshold 1.00E+00*

*DEC:BUS1:SPI:MISOT 1.00E+00*

Query message:

*DEC:BUS1:SPI:MISOT?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:MISOSource

**:DECode:BUS<n>:SPI:MOSISource****Command/Query****DESCRIPTION**

The command selects the MOSI source of the SPI bus.

This query returns the current MOSI source of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:MOSISource <source>

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

<source>:= {C<n>|D<d>|DIS}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

- ◆ DIS means no source selected

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:MOSISource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>|DIS}

**EXAMPLE**

The following command selects the MOSI source of the SPI on bus 1 as C1.

Command message:

*:DECode:BUS1:SPI:MOSISource C1*

*DEC:BUS1:SPI:MOSIS C1*

Query message:

*DEC:BUS1:SPI:MOSIS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:MOSIThreshold

**:DECode:BUS<n>:SPI:MOSIThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the MOSI.

This query returns the current threshold of the MOSI.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:MOSIThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:MOSIThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the MOSI to 1 V on bus 1.

Command message:

*:DECode:BUS1:SPI:MOSIThreshold 1.00E+00  
DEC:BUS1:SPI:MOSIT 1.00E+00*

Query message:

*DEC:BUS1:SPI:MOSIT?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:MOSISource

**:DECode:BUS<n>:SPI:NCSSource**

**Command/Query**

**DESCRIPTION**

The command sets the NCS source of the SPI bus.

This query returns the current NCS source of the SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:NCSSource <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:NCSSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the NCS source of the SPI on bus 1 as C1.

Command message:

*:DECode:BUS1:SPI:NCSSource C1*

*DEC:BUS1:SPI:NCSS C1*

Query message:

*DEC:BUS1:SPI:NCSS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:NCSThreshold

**:DECode:BUS<n>:SPI:NCSThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the NCS on SPI bus.

This query returns the current threshold of the NCS on SPI bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SPI:NCSThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:SPI:NCSThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the NCS on bus 1 to 1 V.

Command message:

*:DECode:BUS1:SPI:NCSThreshold 1.00E+00*

*DEC:BUS1:SPI:NCST 1.00E+00*

Query message:

*DEC:BUS1:SPI:NCST?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SPI:NCSSource



### **:DECode:BUS<n>:UART Commands**

The :DECode:BUS<n>:UART subsystem commands control the UART decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:UART:BAUD**
- ◆ **:DECode:BUS<n>:UART:BITOrder**
- ◆ **:DECode:BUS<n>:UART:DLENgth**
- ◆ **:DECode:BUS<n>:UART:IDLE**
- ◆ **:DECode:BUS<n>:UART:PARity**
- ◆ **:DECode:BUS<n>:UART:RXSource**
- ◆ **:DECode:BUS<n>:UART:**
- ◆ **:DECode:BUS<n>:UART:STOP**
- ◆ **:DECode:BUS<n>:UART:TXSource**
- ◆ **:DECode:BUS<n>:UART:**

**:DECode:BUS<n>:UART:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the UART bus.

This query returns the current baud rate of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:BAUD <baud>

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

<baud>:=  
{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|38400  
bps|57600bps|115200bps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [300, 20000000].

**QUERY SYNTAX**

:DECode:BUS<n>:UART:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|38400  
bps|57600bps|115200bps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the UART to 9600bps on bus 1.

Command message:

```
:DECode:BUS1:UART:BAUD 9600bps  
DEC:BUS1:UART:BAUD 9600bps
```

Query message:

```
DEC:BUS1:UART:BAUD?
```

Response message:

```
9600bps
```

**:DECode:BUS<n>:UART:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order of the UART bus.

This query returns the current bit order of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:BITorder <order>

<n>= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command.

<order>:= {LSB|MSB}

**QUERY SYNTAX**

:DECode:BUS<n>:UART:BITorder?

**RESPONSE FORMAT**

<order>

<order>:= {LSB|MSB}

- ◆ LSB is Least Significant Bit order
- ◆ MSB is Most Significant Bit order

**EXAMPLE**

The following command sets bit order of the UART bus on bus 1 to LSB.

Command message:

*:DECode:BUS1:UART:BITorder LSB*  
*DEC:BUS1:UART:BIT LSB*

Query message:

*DEC:BUS1:UART:BIT?*

Response message:

*LSB*

**:DECode:BUS<n>:UART:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the UART bus.

This query returns the current data length of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:DLENgth <value>

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

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of value is [5, 8].

**QUERY SYNTAX**

:DECode:BUS<n>:UART:DLENgth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the UART to 5 on bus 1.

Command message:

```
:DECode:BUS1:UART:DLENgth 5  
DEC:BUS1:UART:DLEN 5
```

Query message:

```
DEC:BUS1:UART:DLEN?
```

Response message:

```
5
```

**:DECode:BUS<n>:UART:IDLE****Command/Query****DESCRIPTION**

The command sets the idle level of the UART bus.

This query returns the current idle level of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:IDLE <idle>

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

<idle>:= {LOW|HIGH}

**QUERY SYNTAX**

:DECode:BUS<n>:UART:IDLE?

**RESPONSE FORMAT**

<idle>

<idle>:= {LOW|HIGH}

- ◆ LOW means that the idle voltage value is low
- ◆ HIGH means that the idle voltage value is high

**EXAMPLE**

The following command sets the idle level of the UART on bus 1 to low.

Command message:

*:DECode:BUS1:UART:IDLE LOW*  
*DEC:BUS1:UART:IDLE LOW*

Query message:

*DEC:BUS1:UART:IDLE?*

Response message:

*LOW*

**:DECode:BUS<n>:UART:PARity****Command/Query****DESCRIPTION**

The command sets the parity check of the UART bus.

This query returns the current parity check of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:PARity <parity>

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

<parity>:= {NONE|ODD|EVEN|MARK|SPACE}

**QUERY SYNTAX**

:DECode:BUS<n>:UART:PARity?

**RESPONSE FORMAT**

<parity>

<parity>:= {NONE|ODD|EVEN|MARK|SPACE}

**EXAMPLE**

The following command sets the parity check of the UART on bus 1 to NONE.

Command message:

*:DECode:BUS1:UART:PARity NONE*

*DEC:BUS1:UART:PAR NONE*

Query message:

*DEC:BUS1:UART:PAR?*

Response message:

*NONE*

**:DECode:BUS<n>:UART:RXSource****Command/Query****DESCRIPTION**

The command sets the RX source of the UART bus.

This query returns the current RX source of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:RXSource <source>

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

<source>:= {C<n>|D<d>|DIS}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

- ◆ DIS means no source selected

**QUERY SYNTAX**

:DECode:BUS<n>:UART:RXSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>|DIS}

**EXAMPLE**

The following command sets the RX source of the UART on bus 1 as C1.

Command message:

*:DECode:BUS1:UART:RXSource C1*

*DEC:BUS1:UART:RXS C1*

Query message:

*DEC:BUS1:UART:RXS?*

Response message:

*C1*

**:DECode:BUS<n>:UART:RXThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of RX on UART bus.

This query returns the current threshold of RX on UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:RXThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:UART:RXThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the RX to 1 V on bus 1.

Command message:

*:DECode:BUS1:UART:RXThreshold 1.00E+00*

*DEC:BUS1:UART:RXTH 1.00E+00*

Query message:

*DEC:BUS1:UART:RXTH?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:UART:RXSource



**:DECode:BUS<n>:UART:STOP****Command/Query****DESCRIPTION**

The command sets the length of the stop bit on UART bus.

This query returns the current length of the stop bit on UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:STOP <bit>

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

<bit>:= {1|1.5|2}

**QUERY SYNTAX**

:DECode:BUS<n>:UART:STOP?

**RESPONSE FORMAT**

<bit>

<bit>:= {1|1.5|2}

**EXAMPLE**

The following command sets the current length of the stop bit to 1 on bus 1.

Command message:

*:DECode:BUS1:UART:STOP 1*

*DEC:BUS1:UART:STOP 1*

Query message:

*DEC:BUS1:UART:STOP?*

Response message:

*1*

**:DECode:BUS<n>:UART:TXSource****Command/Query****DESCRIPTION**

The command sets the TX source of the UART bus.

This query returns the current TX source of the UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:TXSource <source>

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

<source>:= {C<n>|D<d>|DIS}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

- ◆ DIS means no source selected

**QUERY SYNTAX**

:DECode:BUS<n>:UART:TXSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>|DIS}

**EXAMPLE**

The following command sets the TX source of the UART on bus 1 as C1.

Command message:

```
:DECode:BUS1:UART:TXSource C1
DEC:BUS1:UART:TXS C1
```

Query message:

```
DEC:BUS1:UART:TXS?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:DECode:BUS<n>:UART:

**:DECode:BUS<n>:UART:TXThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of TX on UART bus.

This query returns the current threshold of TX on UART bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:UART:TXThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:UART:TXThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the TX to 1 V on bus 1.

Command message:

*:DECode:BUS1:UART:TXThreshold 1.00E+00  
DEC:BUS1:UART:TXTH 1.00E+00*

Query message:

*DEC:BUS1:UART:TXTH?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:UART:TXSource

**:DECode:BUS<n>:CAN Commands**

The :DECode:BUS<n>:CAN subsystem commands control the CAN decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:CAN:BAUD**
- ◆ **:DECode:BUS<n>:CAN:SOURce**
- ◆ **:DECode:BUS<n>:CAN:THReshold**

**:DECode:BUS<n>:CAN:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the CAN bus.

This query returns the current baud rate of the CAN bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CAN:BAUD <baud>

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

<baud>:=  
{5kbps|10kbps|20kbps|50kbps|100kbps|125kbps|250kbps|500kbps|800kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [5000, 1000000].

**QUERY SYNTAX**

:DECode:BUS<n>:CAN:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{5kbps|10kbps|20kbps|50kbps|100kbps|125kbps|250kbps|500kbps|800kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the CAN on bus 1 to 10kbps.

Command message:

*:DECode:BUS1:CAN:BAUD 10kbps*  
*DEC:BUS1:CAN:BAUD 10kbps*

Query message:

*DEC:BUS1:CAN:BAUD?*

Response message:

*10kbps*

**:DECode:BUS<n>:CAN:SOURce**

### Command/Query

#### DESCRIPTION

The command selects the source of the CAN bus.

This query returns the current source of the CAN bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:CAN:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:CAN:SOURce?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command selects the source of the CAN on bus 1 as C1.

Command message:

*:DECode:BUS1:CAN:SOURce C1*

*DEC:BUS1:CAN:SOUR C1*

Query message:

*DEC:BUS1:CAN:SOUR?*

Response message:

*C1*

#### RELATED COMMANDS

:DECode:BUS<n>:CAN:THReshold

**:DECode:BUS<n>:CAN:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on CAN bus.

This query returns the current threshold of the source on CAN bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CAN:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:CAN:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the CAN bus source to 1 V on bus 1.

Command message:

*:DECode:BUS1:CAN:THReshold 1.00E+00*

*DEC:BUS1:CAN:THR 1.00E+00*

Query message:

*DEC:BUS1:CAN:THR?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:CAN:SOURce

**:DECode:BUS<n>:LIN Commands**

The :DECode:BUS<n>:LIN subsystem commands control the LIN decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:LIN:BAUD**
- ◆ **:DECode:BUS<n>:LIN:SOURce**
- ◆ **:DECode:BUS<n>:LIN:THReshold**



**:DECode:BUS<n>:LIN:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate for the LIN bus.

This query returns the current baud rate for the LIN bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:LIN:BAUD <baud>

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

<baud>:=  
{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|CUSTOM[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [300, 20000000].

**QUERY SYNTAX**

:DECode:BUS<n>:LIN:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|CUSTOM[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate for the LIN to 9600bps on bus 1.

Command message:

*:DECode:BUS1:LIN:BAUD 9600bps*  
*DEC:BUS1:LIN:BAUD 9600bps*

Query message:

*DEC:BUS1:LIN:BAUD?*

Response message:

*9600bps*

**:DECode:BUS<n>:LIN:SOURce**

### Command/Query

#### DESCRIPTION

The command selects the source of the LIN bus.

This query returns the current source of the LIN bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:LIN:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:LIN:SOURce?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command selects the source of the LIN on bus 1 as C1.

Command message:

*:DECode:BUS1:LIN:SOURce C1*

*DEC:BUS1:LIN:SOUR C1*

Query message:

*DEC:BUS1:LIN:SOUR?*

Response message:

*C1*

#### RELATED COMMANDS

:DECode:BUS<n>:LIN:THReshold

**:DECode:BUS<n>:LIN:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on LIN bus.

This query returns the current threshold of the source on LIN bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:LIN:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:DECode:BUS<n>:LIN:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the LIN source to 1 V on bus 1.

Command message:  
*:DECode:BUS1:LIN:THReshold 1.00E+00*  
*DEC:BUS1:LIN:THR 1.00E+00*  
 Query message:  
*DEC:BUS1:LIN:THR?*  
 Response message:  
*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:LIN:SOURce

**:DECode:BUS<n>:FLEXray Commands [Option]**

The :DECode:BUS<n>:FLEXray subsystem commands control the FLEXray decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:FLEXray:BAUD**
- ◆ **:DECode:BUS<n>:FLEXray:SOURce**
- ◆ **:DECode:BUS<n>:FLEXray:THReshold**

**:DECode:BUS<n>:FLEXray:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the Flexray bus.

This query returns the current baud rate of the Flexray bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:FLEXray:BAUD <baud>

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

<baud>:= {2500kbps|5Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1000000, 20000000]

**QUERY SYNTAX**

:DECode:BUS<n>:FLEXray:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:= {2500kbps|5Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the Flexray to 5Mbps on bus 1.

Command message:

*:DECode:BUS1:FLEXray:BAUD 5Mbps*

*DEC:BUS1:FLEX:BAUD 5Mbps*

Query message:

*DEC:BUS1:FLEX:BAUD?*

Response message:

*5Mbps*

**:DECode:BUS<n>:FLEXray:SOURce**

### Command/Query

#### DESCRIPTION

The command selects the source of the Flexray bus.

This query returns the current source of the Flexray bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:FLEXray:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:FLEXray:SOURce?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command selects the source of the Flexray on bus 1 as C1.

Command message:

```
:DECode:BUS1:FLEXray:SOURce C1
DEC:BUS1:FLEX:SOUR C1
```

Query message:

```
DEC:BUS1:FLEX:SOUR?
```

Response message:

```
C1
```

#### RELATED COMMANDS

:DECode:BUS<n>:FLEXray:THReshold

**:DECode:BUS<n>:FLEXray:THReshold****Command/Query****DESCRIPTION**

The command sets the threshold of the source on Flexray bus.

This query returns the current threshold of the source on Flexray bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:FLEXray:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                    | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:FLEXray:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the Flexray source to 1 V on bus 1.

Command message:

```
:DECode:BUS1:FLEXray:THReshold 1.00E+00  
DEC:BUS1:FLEX:THR 1.00E+00
```

Query message:

```
DEC:BUS1:FLEX:THR?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:DECode:BUS<n>:FLEXray:SOURce

**:DECode:BUS<n>:CANFd Commands [Option]**

The :DECode:BUS<n>:CANFd subsystem commands control the CANFD decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:CANFd:BAUDData**
- ◆ **:DECode:BUS<n>:CANFd:BAUDNominal**
- ◆ **:DECode:BUS<n>:CANFd:SOURce**
- ◆ **:DECode:BUS<n>:CANFd:THReshold**



**:DECode:BUS<n>:CANFd:BAUDData****Command/Query****DESCRIPTION**

The command sets the data baud rate of the CAN FD bus.

This query returns the current data baud rate of the CAN FD bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CANFd:BAUDData <baud>

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

<baud>:=  
{500kbps|1Mbps|2Mbps|5Mbps|8Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [100000, 10000000]

**QUERY SYNTAX**

:DECode:BUS<n>:CANFd:BAUDData?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{500kbps|1Mbps|2Mbps|5Mbps|8Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data baud rate of the CAN FD to 500kbps on bus 1.

Command message:

```
:DECode:BUS1:CANFd:BAUDData 500kbps
DEC:BUS1:CANF:BAUDD 500kbps
```

Query message:

```
DEC:BUS1:CANF:BAUDD?
```

Response message:

```
500kbps
```

**:DECode:BUS<n>:CANFd:BAUDNominal****Command/Query****DESCRIPTION**

The command sets the nominal baud rate of the CAN FD bus.

This query returns the current nominal baud rate of the CAN FD bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CANFd:BAUDNominal <baud>

<n>:= {1|2} is attached as a suffix to BUS and defines the bus that is affected by the command.

<baud>:=  
{10kbps|25kbps|50kbps|100kbps|250kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [10000, 1000000]

**QUERY SYNTAX**

:DECode:BUS<n>:CANFd:BAUDNominal?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{10kbps|25kbps|50kbps|100kbps|250kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the nominal baud rate of the CAN FD to 50kbps on bus 1.

Command message:

```
:DECode:BUS1:CANFd:BAUDNominal 50kbps  
DEC:BUS1:CANF:BAUDN 50kbps
```

Query message:

```
DEC:BUS1:CANF:BAUDN?
```

Response message:

```
50kbps
```

**:DECode:BUS<n>:CANFd:SOURce****Command/Query****DESCRIPTION**

The command selects the source of the CAN FD bus.

This query returns the current source of the CAN FD bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CANFd:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:CANFd:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the source of the CAN FD on bus 1 as C1.

Command message:

*:DECode:BUS1:CANFd:SOURce C1*

*DEC:BUS1:CANF:SOUR C1*

Query message:

*DEC:BUS1:CANF:SOUR?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:CANFd:THReshold

**:DECode:BUS<n>:CANFd:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on CAN FD bus.

This query returns the current threshold of the source on CAN FD bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:CANFd:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                    | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:CANFd:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the CAN FD source to 1 V on bus 1.

Command message:

*:DECode:BUS1:CANFd:THReshold 1.00E+00  
DEC:BUS1:CANF:THR 1.00E+0*

Query message:

*DEC:BUS1:CANF:THR?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:CANFd:SOURce

**:DECode:BUS<n>:IIS Commands [Option]**

The :DECode:BUS<n>:IIS subsystem commands control the IIS decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:IIS:ANNotate**
- ◆ **:DECode:BUS<n>:IIS:AVARiant**
- ◆ **:DECode:BUS<n>:IIS:BCLKSource**
- ◆ **:DECode:BUS<n>:IIS:BCLKThreshold**
- ◆ **:DECode:BUS<n>:IIS:BITOrder**
- ◆ **:DECode:BUS<n>:IIS:DLENgth**
- ◆ **:DECode:BUS<n>:IIS:DSource**
- ◆ **:DECode:BUS<n>:IIS:DTHReshold**
- ◆ **:DECode:BUS<n>:IIS:LATChedge**
- ◆ **:DECode:BUS<n>:IIS:LCH**
- ◆ **:DECode:BUS<n>:IIS:SBIT**
- ◆ **:DECode:BUS<n>:IIS:WSSource**
- ◆ **:DECode:BUS<n>:IIS:WSTHreshold**

**:DECode:BUS<n>:IIS:ANNotate****Command/Query****DESCRIPTION**

The command specifies the channel for IIS bus to be annotated.

This query returns the current annotated channel of IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:ANNotate <type>

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

<type>:= {ALL|LEFT|RIGHT}

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:ANNotate?

**RESPONSE FORMAT**

<type>

<type>:= {ALL|LEFT|RIGHT}

**EXAMPLE**

The following command annotates all the channels of IIS on bus 1.

Command message:

*:DECode:BUS1:IIS:ANNotate ALL*

*DEC:BUS1:IIS:ANN ALL*

Query message:

*DEC:BUS1:IIS:ANN?*

Response message:

*ALL*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:LCH

**:DECode:BUS<n>:IIS:AVARiant**

**Command/Query**

**DESCRIPTION**

The command selects the audio variant for IIS bus.

This query returns the current audio variant for IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:AVARiant <type>

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

<type>:= {I2S|LJ|RJ}

- ◆ I2S justified.
- ◆ LJ is left justified.
- ◆ RL is right justified.

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:AVARiant?

**RESPONSE FORMAT**

<type>

<type>:= {I2S|LJ|RJ}

**EXAMPLE**

The following command sets the audio variable of the IIS on bus 1 to RJ.

Command message:

*:DECode:BUS1:IIS:AVARiant RJ*  
*DEC:BUS1:IIS:AVAR RJ*

Query message:

*DEC:BUS1:IIS:AVAR?*

Response message:

*RJ*

**:DECode:BUS<n>:IIS:BCLKSource****Command/Query****DESCRIPTION**

The command selects the BCLK source of the IIS bus.

This query returns the current BCLK source of the IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:BCLKSource <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:BCLKSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the BCLK source of IIS on bus 1 as C1.

Command message:

*:DECode:BUS1:IIS:BCLKSource C1*

*DEC:BUS1:IIS:BCLKS C1*

Query message:

*DEC:BUS1:IIS:BCLKS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:BCLKThreshold



**:DECode:BUS<n>:IIS:BCLKThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the BCLK on IIS bus.

This query returns the current threshold of the BCLK on IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:BCLKThreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 * \text{vertical\_scale} - \text{vertical\_offset}, 4.26 * \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 * \text{vertical\_scale} - \text{vertical\_offset}, 4.5 * \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 * \text{vertical\_scale} - \text{vertical\_offset}, 4.1 * \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:BCLKThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the BCLK to 1 V on bus 1.

Command message:

*:DECode:BUS1:IIS:BCLKThreshold 1.00E+00  
DEC:BUS1:IIS:BCLKT 1.00E+00*

Query message:

*DEC:BUS1:IIS:BCLKT?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:BCLKSource

**:DECode:BUS<n>:IIS:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order for the IIS bus.

This query returns the current bit order for the IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:BITorder <order>

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

<order>:= {LSB|MSB}

- ◆ LSB is Least Significant Bit.
- ◆ MSB is Most Significant Bit.

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:BITorder?

**RESPONSE FORMAT**

<order>

<order>:= {LSB|MSB}

**EXAMPLE**

The following command sets bit order for the IIS on bus 1 to LSB.

Command message:

*:DECode:BUS1:IIS:BITorder LSB*  
*DEC:BUS1:IIS:BIT LSB*

Query message:

*DEC:BUS1:IIS:BIT?*

Response message:

*LSB*

**:DECode:BUS<n>:IIS:DLENgth****Command/Query****DESCRIPTION**

The command sets the data bits for the IIS bus.

This query returns the current data bits for the IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:DLENgth <value>

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

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:DLENgth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data bits for the IIS to 5 on bus 1.

Command message:

*:DECode:BUS1:IIS:DLENgth 5*  
*DEC:BUS1:IIS:DLEN 5*

Query message:

*DEC:BUS1:IIS:DLEN?*

Response message:

*5*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:SBIT

**:DECode:BUS<n>:IIS:DSource**

**Command/Query**

**DESCRIPTION**

The command selects the data source of the IIS bus.

This query returns the current data source of the IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:DSource <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:DSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the data source of the IIS bus on bus 1 as C1.

Command message:

*:DECode:BUS1:IIS:DSource C1*

*DEC:BUS1:IIS:DS C1*

Query message:

*DEC:BUS1:IIS:DS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:DTHReshold

**:DECode:BUS<n>:IIS:DTHReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the data source on IIS bus.

This query returns the current threshold of the data source on IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:DTHReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 * \text{vertical\_scale} - \text{vertical\_offset}, 4.26 * \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 * \text{vertical\_scale} - \text{vertical\_offset}, 4.5 * \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 * \text{vertical\_scale} - \text{vertical\_offset}, 4.1 * \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:DTHReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the data source to 1 V on bus 1.

Command message:

*:DECode:BUS1:IIS:DTHReshold 1.00E+00  
DEC:BUS1:IIS:DTHR 1.00E+00*

Query message:

*DEC:BUS1:IIS:DTHR?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:DSource

**:DECode:BUS<n>:IIS:LATChedge****Command/Query****DESCRIPTION**

The command selects the sampling edge of BCLK on IIS bus.

This query returns the sampling edge of BCLK on IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:LATChedge <slope>

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

<slope>:= {RISing|FALLing}

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:LATChedge?

**RESPONSE FORMAT**

<slope>

<slope>:= {RISing|FALLing}

- ◆ RISing selects the rising edge.
- ◆ FALLing selects the falling edge.

**EXAMPLE**

The following command sets the sampling edge of BCLK on bus 1 to RISing.

Command message:

```
:DECode:BUS1:IIS:LATChedge RISing
DEC:BUS1:IIS:LATC RIS
```

Query message:

```
DEC:BUS1:IIS:LATC?
```

Response message:

```
RISing
```

**:DECode:BUS<n>:IIS:LCH****Command/Query****DESCRIPTION**

The command selects the level of the left channel.

This query returns the current level of the left channel.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:LCH <left>

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

<left>:= {LOW|HIGH}

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:LCH?

**RESPONSE FORMAT**

<left>

<left>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the left channel on bus 1 to LOW.

Command message:

*:DECode:BUS1:IIS:LCH LOW*

*DEC:BUS1:IIS:LCH LOW*

Query message:

*DEC:BUS1:IIS:LCH?*

Response message:

*LOW*

**:DECode:BUS<n>:IIS:SBIT****Command/Query****DESCRIPTION**

The command sets the start bit of the data.

This query returns the start bit of the data.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:SBIT <value>

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

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 31].

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:SBIT?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of the data bit to 1 on bus 1.

Command message:

*:DECode:BUS1:IIS:SBIT 1*

*:DEC:BUS1:IIS:SBIT 1*

Query message:

*DEC:BUS1:IIS:SBIT?*

Response message:

*1*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:DLENgth



**:DECode:BUS<n>:IIS:WSSource**

**Command/Query**

**DESCRIPTION**

The command selects the WS source of the IIS bus.

This query returns the current WS source of the IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:WSSource <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:WSSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the WS source of the IIS bus on bus 1 as C1.

Command message:

*:DECode:BUS1:IIS:WSSource C1*

*DEC:BUS1:IIS:WSS C1*

Query message:

*DEC:BUS1:IIS:WSS?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:WSTHreshold

**:DECode:BUS<n>:IIS:WSTHreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the WS on IIS bus.

This query returns the current threshold of the WS on IIS bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:IIS:WSTHreshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:IIS:WSTHreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the WS to 1 V on bus 1.

Command message:

*:DECode:BUS1:IIS:WSTHreshold 1.00E+00  
DEC:BUS1:IIS:WSTH 1.00E+00*

Query message:

*DEC:BUS1:IIS:WSTH?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:IIS:WSSource

### **:DECode:BUS<n>:M1553 Commands [Option]**

The :DECode:BUS<n>:M1553 subsystem commands control the M1553 decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:M1553:LTHReshold**
- ◆ **:DECode:BUS<n>:M1553:SOURce**
- ◆ **:DECode:BUS<n>:M1553:UTHReshold**

**:DECode:BUS<n>:M1553:LTHReshold**

**Command/Query**

**DESCRIPTION**

The command sets the lower threshold of the M1553 source.

This query returns the current lower threshold of the M1553 source.

**COMMAND SYNTAX**

:DECode:BUS<n>:M1553:LTHReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The lower threshold value cannot be greater than the upper threshold value set by the command

:DECode:BUS<n>:M1553:UTHReshold.

**QUERY SYNTAX**

:DECode:BUS<n>:M1553:LTHReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower threshold of the M1553 source to 1 V on bus 1.

Command message:

*:DECode:BUS1:M1553:LTHReshold 1.00E+00*

*DEC:BUS1:M1553:LTHR 1.00E+00*

Query message:

*DEC:BUS1:M1553:LTHR?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:M1553:SOURce

:DECode:BUS<n>:M1553:UTHReshold

**:DECode:BUS<n>:M1553:SOURce****Command/Query****DESCRIPTION**

The command selects the source of the M1553 bus.

This query returns the current source of the M1553 bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:M1553:SOURce <source>

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

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DECode:BUS<n>:M1553:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command selects the source of the M1553 as C1 on bus 1.

Command message:

*:DECode:BUS1:M1553:SOURce C1*

*DEC:BUS1:M1553:SOUR C1*

Query message:

*DEC:BUS1:M1553:SOUR?*

Response message:

*C1*

**RELATED COMMANDS**

:DECode:BUS<n>:M1553:UTHReshold

:DECode:BUS<n>:M1553:LTHReshold

**:DECode:BUS<n>:M1553:UTHReshold**

**Command/Query**

**DESCRIPTION**

The command sets the upper threshold of the M1553 source.

This query returns the current upper threshold of the M1553 source.

**COMMAND SYNTAX**

:DECode:BUS<n>:M1553:UTHReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 * \text{vertical\_scale} - \text{vertical\_offset}, 4.26 * \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 * \text{vertical\_scale} - \text{vertical\_offset}, 4.5 * \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 * \text{vertical\_scale} - \text{vertical\_offset}, 4.1 * \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The upper threshold value cannot be less than the lower threshold value set by the command

:DECode:BUS<n>:M1553:LTHReshold.

**QUERY SYNTAX**

:DECode:BUS<n>:M1553:UTHReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper threshold of the M1553 bus source to 2 V on bus 1.

Command message:

*:DECode:BUS1:M1553:UTHReshold 2.00E+00  
DEC:BUS1:M1553:UTHR 2.00E+00*

Query message:

*DEC:BUS1:M1553:UTHR?*

Response message:

*2.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:M1553:SOURce

:DECode:BUS<n>:M1553:LTHReshold

### **:DECode:BUS<n>:SENT Commands [Option]**

The :DECode:BUS<n>:SENT subsystem commands control the SENT decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:SENT:SOURce**
- ◆ **:DECode:BUS<n>:SENT:THReshold**
- ◆ **:DECode:BUS<n>:SENT:FORMat**
- ◆ **:DECode:BUS<n>:SENT:CLOCK**
- ◆ **:DECode:BUS<n>:SENT:TOLerance**
- ◆ **:DECode:BUS<n>:SENT:IDLE**
- ◆ **:DECode:BUS<n>:SENT:LENGth**
- ◆ **:DECode:BUS<n>:SENT:CRC**
- ◆ **:DECode:BUS<n>:SENT:PPULse**

**:DECode:BUS<n>:SENT:SOURce**

### Command/Query

#### DESCRIPTION

The command selects the source of the SENT bus.

This query returns the current source of the SENT bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:SENT:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:SENT:SOURce?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command selects the source of the SENT as C1 on bus 1.

Command message:

*:DECode:BUS1:SENT:SOURce C1*

*DEC:BUS1:SENT:SOUR C1*

Query message:

*DEC:BUS1:SENT:SOUR?*

Response message:

*C1*

#### RELATED COMMANDS

:DECode:BUS<n>:SENT:THReshold



**:DECode:BUS<n>:SENT:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on SENT bus.

This query returns the current threshold of the source on SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model                                  | Value Range   |
|--|---|
| SDS7000A                               | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L    | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the SENT bus source to 1 V on bus 1.

Command message:

*:DECode:BUS1:SENT:THReshold 1.00E+00  
DEC:BUS1:SENT:THR 1.00E+00*

Query message:

*DEC:BUS1:SENT:THR?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:DECode:BUS<n>:SENT:SOURce

**:DECode:BUS<n>:SENT:FORMat**

**Command/Query**

**DESCRIPTION**

The command selects the message format of the SENT bus.

This query returns the message format of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:FORMat <format>

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

<format>:= {NIBBles|FSIGNal|SSERial|ESERial}

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:FORMat?

**RESPONSE FORMAT**

<format>

<format>:= {NIBBles|FSIGNal|SSERial|ESERial}

**EXAMPLE**

The following command selects the message format of the SENT bus of the bus 1 as NIBBles.

Command message:

*:DECode:BUS1:SENT:FORMat NIBBles*

*DEC:BUS1:SENT:FORM NIBB*

Query message:

*DEC:BUS1:SENT:FORM?*

Response message:

*NIBBles*

**:DECode:BUS<n>:SENT:CLOCK****Command/Query****DESCRIPTION**

The command sets the clock period (tick) time of the SENT bus.

**COMMAND SYNTAX**

This query returns the current clock period of the SENT bus.  
:DECode:BUS<n>:SENT:CLOCK <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [500E-09, 300E-06]

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:CLOCK?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the clock cycle of SENT bus on the bus 1 to 1us.

Command message:

*:DECode:BUS1:SENT:CLOCK 1.00E-06*  
*DEC:BUS1:SENT:CLOC 1.00E-06*

Query message:

*DEC:BUS1:SENT:CLOC?*

Response message:

*1.00E-06*

**:DECode:BUS<n>:SENT:TOLerance****Command/Query****DESCRIPTION**

The command sets the clock percent tolerance of the SENT bus.

This query returns the current clock tolerance of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:TOLerance <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 25].

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:TOLerance?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the tolerance of the SENT on bus 1 to 5%.

Command message:

```
:DECode:BUS1:SENT:TOLerance 5  
DEC:BUS1:SENT:TOL 5
```

Query message:

```
DEC:BUS1:SENT:TOL?
```

Response message:

```
5
```

**:DECode:BUS<n>:SENT:IDLE****Command/Query****DESCRIPTION**

The command sets the idle level of the SENT bus.

The query returns the current idle level of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:IDLE <idle>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<idle>:= {LOW|HIGH}

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:IDLE?

**RESPONSE FORMAT**

<idle>

<idle>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the idle level of the SENT bus of the bus 1 as low.

Command message:

*:DECode:BUS1:SENT:IDLE LOW*

*DEC:BUS1:SENT:IDLE LOW*

Query message:

*DEC:BUS1:SENT:IDLE?*

Response message:

*LOW*

**:DECode:BUS<n>:SENT:LENGth**

**Command/Query**

**DESCRIPTION**

The command sets the number of nibbles of the SENT bus.

This query returns the current number of nibbles of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:LENGth <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [3, 8].

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:LENGth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the Number of nibbles of the SENT on bus 1 to 5.

Command message:

*:DECode:BUS1:SENT:LENGth 5*  
*DEC:BUS1:SENT:LENG 5*

Query message:

*DEC:BUS1:SENT:LENG?*

Response message:

*5*

**:DECode:BUS<n>:SENT:CRc****Command/Query****DESCRIPTION**

The command sets the CRC format of the SENT bus.

The query returns the CRC format of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:CRc <state>

<state>:= {OFF|ON}

ON sets to 2010 CRC format.

OFF sets to 2008 CRC format.

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:CRc?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command sets to 2010 CRC format of the SENT bus on the bus 1.

Command message:

*:DECode:BUS1:SENT:CRc ON*

*DEC:BUS1:SENT:CRc ON*

Query message:

*DEC:BUS1:SENT:CRc?*

Response message:

*ON*

---

**:DECode:BUS<n>:SENT:PPULse**

**Command/Query**

**DESCRIPTION**

The command sets the state of pause pulse of the SENT bus.

The query returns the current state of pause pulse of the SENT bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:SENT:PPULse <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:DECode:BUS<n>:SENT:PPULse?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command sets the state of pause of the SENT bus on the bus 1 as ON.

Command message:

*:DECode:BUS1:SENT:PPULse ON*  
*DEC:BUS1:SENT:PPUL ON*

Query message:

*DEC:BUS1:SENT:PPUL?*

Response message:

*ON*



**:DECode:BUS<n>:MANChester Commands [Option]**

The :DECode:BUS<n>:MANChester subsystem commands control the MANChester decode settings of the specified bus.

- ◆ **:DECode:BUS<n>:MANChester:SOURce**
- ◆ **:DECode:BUS<n>:MANChester:THReshold**
- ◆ **:DECode:BUS<n>:MANChester:BAUD**
- ◆ **:DECode:BUS<n>:MANChester:POLarity**
- ◆ **:DECode:BUS<n>:MANChester:IDLE**
- ◆ **:DECode:BUS<n>:MANChester:IBITs**
- ◆ **:DECode:BUS<n>:MANChester:STARt**
- ◆ **:DECode:BUS<n>:MANChester:SSIZe**
- ◆ **:DECode:BUS<n>:MANChester:HSIZe**
- ◆ **:DECode:BUS<n>:MANChester:TSIZe**
- ◆ **:DECode:BUS<n>:MANChester:WSIZe**
- ◆ **:DECode:BUS<n>:MANChester:DSIZe**
- ◆ **:DECode:BUS<n>:MANChester:DISPlay**
- ◆ **:DECode:BUS<n>:MANChester:BITorder**

**:DECode:BUS<n>:MANChester:SOURce**

### Command/Query

#### DESCRIPTION

The command selects the source of the Manchester bus.

This query returns the current source of the Manchester bus.

#### COMMAND SYNTAX

:DECode:BUS<n>:MANChester:SOURce <source>

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

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### QUERY SYNTAX

:DECode:BUS<n>:MANChester:SOURce?

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|D<d>}

#### EXAMPLE

The following command selects the source of the Manchester as C1 on bus 1.

Command message:

```
:DECode:BUS1:MANChester:SOURce C1
DEC:BUS1:MANC:SOUR C1
```

Query message:

```
DEC:BUS1:MANC:SOUR?
```

Response message:

```
C1
```

#### RELATED COMMANDS

:DECode:BUS<n>:MANChester:THReshold

**:DECode:BUS<n>:MANChesTer:THReshold****Command/Query****DESCRIPTION**

The command sets the threshold of the source on Manchester bus.

This query returns the current threshold of the source on Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChesTer:THReshold <value>

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

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model                               | Value Range   |
|-------------------------------------|---|
| SDS7000A                            | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS2000X HD             | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:DECode:BUS<n>:MANChesTer:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the Manchester bus source to 1 V on bus 1.

Command message:

```
:DECode:BUS1:MANChesTer:THReshold 1.00E+00
DEC:BUS1:MANC:THR 1.00E+00
```

Query message:

```
DEC:BUS1:MANC:THR?
```

Response message:

```
1.00E+00
```

**RELATED COMMANDS**

:DECode:BUS<n>:MANChesTer:SOURce

**:DECode:BUS<n>:MANChester:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate for the Manchester bus.

This query returns the current baud rate for the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:BAUD <baud>

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

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [500, 5000000].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:BAUD?

**RESPONSE FORMAT**

<baud>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate for the Manchester to 9600bps on bus 1.

Command message:

*:DECode:BUS1:MANChester:BAUD 9600*  
*DEC:BUS1:MANC:BAUD 9600*

Query message:

*DEC:BUS1:MANC:BAUD?*

Response message:

*9600*

**:DECode:BUS<n>:MANChesTer:POLarity****Command/Query****DESCRIPTION**

The command sets the signal's logic type of the Manchester bus.

The query returns the current polarity of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChesTer:POLarity <polar>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<polar>:= {RISing|FALLing}

- ◆ RISing indicates that rising edge is used to encode a bit value of logic 1.
- ◆ FALLing indicates that falling edge is used to encode a bit value of logic 1.

**QUERY SYNTAX**

:DECode:BUS<n>:MANChesTer:POLarity?

**RESPONSE FORMAT**

<polar>

<polar>:= {RISing|FALLing}

**EXAMPLE**

The following command encodes the rising edge of the Manchester bus of the bus 1 as logic 1.

Command message:

*:DECode:BUS1:MANChesTer:POLarity RISing  
DEC:BUS1:MANC:POL RIS*

Query message:

*DEC:BUS1:MANC:POL?*

Response message:

*RISing*

**:DECode:BUS<n>:MANChester:IDLE****Command/Query****DESCRIPTION**

The command sets the idle level of the Manchester bus.

The query returns the current idle level of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:IDLE <idle>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<idle>:= {LOW|HIGH}

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:IDLE?

**RESPONSE FORMAT**

<idle>

<idle>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the idle level of the Manchester bus of the bus 1 as LOW.

Command message:

*:DECode:BUS1:MANChester:IDLE LOW*  
*DEC:BUS1:MANC:IDLE LOW*

Query message:

*DEC:BUS1:MANC:IDLE?*

Response message:

*LOW*

**:DECode:BUS<n>:MANChester:IBITs****Command/Query****DESCRIPTION**

The command sets the idle bits of the Manchester bus.

This query returns the current idle bits of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:IBITs <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [2, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:IBITs?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the idle bits of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:IBITs 5*  
*DEC:BUS1:MANC:IBIT 5*

Query message:

*DEC:BUS1:MANC:IBIT?*

Response message:

*5*

**:DECode:BUS<n>:MANChester:STARt****Command/Query****DESCRIPTION**

The command sets the start edge of the Manchester bus.

This query returns the current start edge of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:STARt <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:STARt?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the start edge of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:STARt 5*  
*DEC:BUS1:MANC:STAR 5*

Query message:

*DEC:BUS1:MANC:STAR?*

Response message:

*5*



**:DECode:BUS<n>:MANChester:SSIZe****Command/Query****DESCRIPTION**

The command sets the sync size of the Manchester bus.

This query returns the current sync size of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:SSIZe <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:SSIZe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the sync size of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:SSIZe 5*  
*DEC:BUS1:MANC:SSIZ 5*

Query message:

*DEC:BUS1:MANC:SSIZ?*

Response message:

*5*

**:DECode:BUS<n>:MANChester:HSIZe**

**Command/Query**

**DESCRIPTION**

The command sets the header size of the Manchester bus.

This query returns the current header size of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:HSIZe <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:HSIZe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the header size of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:HSIZe 5*  
*DEC:BUS1:MANC:HSIZ 5*

Query message:

*DEC:BUS1:MANC:HSIZ?*

Response message:

*5*

**RELATED COMMANDS**

:DECode:BUS<n>:MANChester:DISPlay

**:DECode:BUS<n>:MANChester:TSIZe****Command/Query****DESCRIPTION**

The command sets the trailer size of the Manchester bus.

This query returns the current trailer size of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:TSIZe <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 32].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:TSIZe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the trailer size of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:TSIZe 5*  
*DEC:BUS1:MANC:TSIZ 5*

Query message:

*DEC:BUS1:MANC:TSIZ?*

Response message:

*5*

**RELATED COMMANDS**

:DECode:BUS<n>:MANChester:DISPlay

**:DECode:BUS<n>:MANChester:WSize**

**Command/Query**

**DESCRIPTION**

The command sets the word size of the Manchester bus.

This query returns the current word size of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:WSize <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [2, 8].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:WSize?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the word size of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:WSize 5*  
*DEC:BUS1:MANC:WSIZ 5*

Query message:

*DEC:BUS1:MANC:WSIZ?*

Response message:

*5*

**RELATED COMMANDS**

:DECode:BUS<n>:MANChester:DISPlay

**:DECode:BUS<n>:MANChester:DSIZe****Command/Query****DESCRIPTION**

The command sets the data word length of the Manchester bus.

This query returns the current data word length of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:DSIZe <value>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 255].

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:DSIZe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data bits of the Manchester on bus 1 to 5.

Command message:

*:DECode:BUS1:MANChester:DSIZe 5*

*DEC:BUS1:MANC:DSIZ 5*

Query message:

*DEC:BUS1:MANC:DSIZ?*

Response message:

*5*

**RELATED COMMANDS**

:DECode:BUS<n>:MANChester:DISPlay

**:DECode:BUS<n>:MANChester:DISPlay****Command/Query****DESCRIPTION**

The command sets the display format of the Manchester bus.

The query returns the current display format of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:DISPlay <format>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<format>:= {WORD|BIT}

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:DISPlay?

**RESPONSE FORMAT**

<format>

<format>:= {WORD|BIT}

**EXAMPLE**

The following command sets the display format of the Manchester bus of the bus 1 as WORD.

Command message:

*:DECode:BUS1:MANChester:DISPlay WORD*  
*DEC:BUS1:MANC:DISP WORD*

Query message:

*DEC:BUS1:MANC:DISP?*

Response message:

*WORD*

**:DECode:BUS<n>:MANChester:BITOrder****Command/Query****DESCRIPTION**

The command sets the bit order of the Manchester bus.

The query returns the current bit order of the Manchester bus.

**COMMAND SYNTAX**

:DECode:BUS<n>:MANChester:BITOrder <order>

<n>:= {1|2}, is attached as a suffix to BUS and defines the bus that is affected by the command

<order>:= {LSB|MSB}

**QUERY SYNTAX**

:DECode:BUS<n>:MANChester:BITOrder?

**RESPONSE FORMAT**

<order>

<order>:= {LSB|MSB}

**EXAMPLE**

The following command sets the bit order of the Manchester bus of the bus 1 as MSB.

Command message:

*:DECode:BUS1:MANChester:BITOrder MSB*

*DEC:BUS1:MANC:BIT MSB*

Query message:

*DEC:BUS1:MANC:BIT?*

Response message:

*MSB*

## DIGital Commands [Option]

The :DIGital subsystem commands control the viewing of digital channels. They also control threshold settings for groups of digital channels.

- ◆ :DIGital
- ◆ :DIGital:ACTive
- ◆ :DIGital:BUS<n>:DISPlay
- ◆ :DIGital:BUS<n>:DEFault
- ◆ :DIGital:BUS<n>:FORMat
- ◆ :DIGital:BUS<n>:MAP
- ◆ :DIGital:D<d>
- ◆ :DIGital:HEIGht
- ◆ :DIGital:LABel<d>
- ◆ :DIGital:POINts
- ◆ :DIGital:POSition
- ◆ :DIGital:SKEW
- ◆ :DIGital:SRATe
- ◆ :DIGital:THReshold<n>



**:DIGital****Command/Query****DESCRIPTION**

The command set the switch of the digital.

This query returns the current state of the digital.

**COMMAND SYNTAX**

:DIGital <state>

<state>:= {ON|OFF}

- ◆ ON enables the channel.
- ◆ OFF disables the channel.

**QUERY SYNTAX**

:DIGital?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables digital function.

Command message:

*:DIGital ON*

*DIG ON*

Query message:

*DIG?*

Response message:

*ON*

**:DIGital:ACTive****Command/Query****DESCRIPTION**

This command activates the specified digital channel.

This query returns the active digital channel.

**COMMAND SYNTAX**

:DIGital:ACTive <digital>

<digital>:= {D<d>}

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DIGital:ACTive?

**RESPONSE FORMAT**

<digital>

<digital>:= {D<d>}

**EXAMPLE**

The following command selects the digital channel 5 waveform.

Command message:

*:DIGital:ACTive D5*

*DIG:ACT D5*

Query message:

*DIG:ACT?*

Response message:

*D5*

**:DIGital:BUS<n>:DISPlay****Command/Query****DESCRIPTION**

The command sets the display of the specified digital bus.

This query returns the current display of the specified digital bus.

**COMMAND SYNTAX**

:DIGital:BUS<n>:DISPlay <state>

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

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DIGital:BUS<n>:DISPlay?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

- ◆ ON displays the selected bus.
- ◆ OFF removes the selected bus from the display.

**EXAMPLE**

The following command sets digital bus 1 on.

Command message:

*:DIGital:BUS1:DISPlay ON*  
*DIG:BUS1:DISP ON*

Query message:

*DIG:BUS1:DISP?*

Response message:

*ON*

**:DIGital:BUS<n>:DEFault****Command****DESCRIPTION**

This command resets the digital channel bus bit order

**COMMAND SYNTAX**

:DIGital:BUS<n>:DEFault

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

**EXAMPLE**

The following command resets the digital channel bus1 data.

Command message:

*:DIGital:BUS1:DEFault*

*DIG:BUS1:DEF*

**RELATED COMMANDS**

:DIGital:BUS<n>:MAP

**:DIGital:BUS<n>:FORMat****Command/Query****DESCRIPTION**

The command selects the display format of the specified digital bus.

This query returns the current display format of the specified digital bus.

**COMMAND SYNTAX**

:DIGital:BUS<n>:FORMat <format>

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

<format>:= {BINary|DECimal|UDECimal|HEX}

- ◆ BINary presents the decoded data in binary format
- ◆ DECimal presents the decoded data in decimal format
- ◆ UDECimal presents the decoded data in unsigned decimal format
- ◆ HEX presents the decoded data in hexadecimal format

**QUERY SYNTAX**

:DIGital:BUS<n>:FORMat?

**RESPONSE FORMAT**

<format>

<format>:= {BINary|DECimal|UDECimal|HEX }

**EXAMPLE**

The following command selects the display format of the digital bus 1 to HEX.

Command message:

*:DIGital:BUS1:FORMat HEX*

*DIG:BUS1:FORM HEX*

Query message:

*DIG:BUS1:FORM?*

Response message:

*HEX*

**:DIGital:BUS<n>:MAP****Command/Query****DESCRIPTION**

The command sets the bit order of each digital channel in the digital bus and the bit width of the digital bus.

The query returns the current digital bus data composition in the LSB order.

**COMMAND SYNTAX**

:DIGital:BUS<n>:MAP <source>[...[,<source>]]

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

<source>:= {D<d>}

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**Note:**

- It will synchronously set the bit width of the digital bus, which is determined by the number of parameters.
- Use the command :DIGital:BUS<n>:DEFault to reset the bit sequence to d0-d15 according to the current digital bus bit width.

**QUERY SYNTAX**

:DIGital:BUS<n>:MAP?

**RESPONSE FORMAT**

<source>[...[,<source>]]

<source>:= {D<d>}

**EXAMPLE**

The following command the data of the digital bus 1 to D0,D3,D7,D15.

Command message:

*:DIGital:BUS1:MAP D0,D3,D7,D15*

*DIG:BUS1:MAP D0,D3,D7,D15*

Query message:

*DIG:BUS1:MAP?*

Response message:

*D0,D3,D7,D15*

**RELATED COMMANDS**

:DIGital:BUS<n>:DEFault

:DIGital:D<d>

**:DIGital:D<d>****Command/Query****DESCRIPTION**

This command enables or disables the specified digital channel.

This query returns the switch of the specified digital channel.

**COMMAND SYNTAX**

:DIGital:D<d> <state>

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {ON|OFF}

- ◆ ON enables the specified digital channel.
- ◆ OFF disables the specified digital channel.

**QUERY SYNTAX**

:DIGital:D<d>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command closes the digital channel 5.

Command message:

*:DIGital:D5 OFF*  
*DIG:D5 OFF*

Query message:

*DIG:D5?*

Response message:

*OFF*

**RELATED COMMANDS**

:DIGital

**:DIGital:HEIGht****Command/Query****DESCRIPTION**

This command sets the height of digital channel waveform display.

This query returns the height of digital channel waveform display.

**COMMAND SYNTAX**

:DIGital:HEIGht <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. This value indicates the number of divisions occupied by the digital waveform in the vertical direction when the waveform area is not compressed.

The range of the value is [4.00E+00, 8.00E+00].

**QUERY SYNTAX**

:DIGital:HEIGht?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the height of the digital channel display area to 6 div.

Command message:

*:DIGital:HEIGht 6.00E+00*

*DIG:HEIG 6.00E+00*

Query message:

*DIG:HEIG?*

Response message:

*6.00E+00*

**RELATED COMMANDS**

:DIGital:POSition



**:DIGital:LABel<d>****Command/Query****DESCRIPTION**

This command sets the label text of the selected digital channel.

This query returns the current label text of the selected digital channel.

**COMMAND SYNTAX**

:DIGital:LABel<d> <string>

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

<string>:= Quoted string of ASCII text. The length of the string is limited to 8.

**QUERY SYNTAX**

:DIGital:LABel<d>?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the label name of the digital channel 15 to "IIC\_DATA".

Command message:

*:DIGital:LABel15 "IIC\_DATA"*

*DIG:LAB15 "IIC\_DATA"*

Query message:

*DIG:LAB15?*

Response message:

*"IIC\_DATA"*

**RELATED COMMANDS**

:DIGital:D<d>

**:DIGital:POINts****Query****DESCRIPTION**

This query returns the number of sampling points of the digital channel.

**QUERY SYNTAX**

:DIGital:POINts?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command query returns the number of sampling points of the digital channel.

Query message:

*DIG:POIN?*

Response message:

*6.25E+02*

**RELATED COMMANDS**

:DIGital:SRATe

**:DIGital:POSition****Command/Query****DESCRIPTION**

The command sets the position of the digital channel waveform display.

The query returns the position of the digital channel waveform display.

**COMMAND SYNTAX**

:DIGital:POSition <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. This value indicates the number of divisions the digital waveform moves from top to bottom of the waveform area when the waveform area is not compressed

**Note:**

The range of legal values varies with the number of digital channels displayed.

**QUERY SYNTAX**

:DIGital:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the position of the digital channel display area to 4 div when the digital channel height is 4.

Command message:

*:DIGital:POSition 4.00E+00*

*DIG:POS 4.00E+00*

Query message:

*DIG:POS?*

Response message:

*4.00E+00*

**RELATED COMMANDS**

:DIGital:HEIGHt

**:DIGital:SKEW****Command/Query****DESCRIPTION**

This command sets the skew of the digital channel.

This query returns the current skew of the digital channel.

**COMMAND SYNTAX**

:DIGital:SKEW <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value is [-1.00E-07, 1.00E-07].

**QUERY SYNTAX**

:DIGital:SKEW?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the skew of the digital channel to 100 ns.

Command message:

*:DIGital:SKEW 1.00E-07*

*DIG:SKEW 1.00E-07*

Query message:

*DIG:SKEW?*

Response message:

*1.00E-07*

**:DIGital:SRATe****Query****DESCRIPTION**

This command query returns the sampling rate of the digital channel.

**QUERY SYNTAX**

:DIGital:SRATe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command query returns the sampling rate of the digital channel.

Query message:

*DIG:SRAT?*

Response message:

*1.25E+09*

**:DIGital:THReshold<n>**

**Command/Query**

**DESCRIPTION**

This command sets the threshold value of the digital channel group.

This query returns the threshold value of the digital channel group.

**COMMAND SYNTAX**

:DIGital:THReshold<n> <type>

<n>:= {1|2}

- ◆ 1 means D0-D7
- ◆ 2 means D8-D15

<type>:=

{TTL|CMOS|LVCMOS33|LVCMOS25|CUSTOm[,<value>]}

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range    |
|--|----------------|
| SDS7000A<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | [-1E+01,1E+01] |
| SDS1000X HD<br>SDS800X HD  | [-8E-1, 8E-1]  |

**QUERY SYNTAX**

:DIGital:THReshold<n>?

**RESPONSE FORMAT**

<type>

<type>:=

{TTL|CMOS|LVCMOS33|LVCMOS25|CUSTOm[,<value>]}

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold value of D0-D7 to CMOS.

Command message:

*:DIGital:THReshold1 CMOS*

*DIG:THR1 CMOS*

Query message:

*DIG:THR1?*

Response message:

*CMOS*

## DISPlay Commands

The :DISPlay subsystem commands control waveforms and screen displays.

- ◆ :DISPlay:AXIS
- ◆ :DISPlay:AXIS:MODE
- ◆ :DISPlay:AXIS:POSition
- ◆ :DISPlay:BACKlight
- ◆ :DISPlay:CLEAr
- ◆ :DISPlay:COLor
- ◆ :DISPlay:GRATicule
- ◆ :DISPlay:GRIDstyle
- ◆ :DISPlay:HIDemenu
- ◆ :DISPlay:INTensity
- ◆ :DISPlay:MENU
- ◆ :DISPlay:MENU:HIDE
- ◆ :DISPlay:PERStistence
- ◆ :DISPlay:TRANsparence
- ◆ :DISPlay:TYPE



**:DISPlay:AXIS****Command/Query****DESCRIPTION**

The command sets the display of the axis label.

The query returns the current status of the axis label.

**COMMAND SYNTAX**

:DISPlay:AXIS <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DISPlay:AXIS?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the axis label.

Command message:

*:DISPlay:AXIS ON*

*DISP:AXIS ON*

Query message:

*DISP:AXIS?*

Response message:

*ON*

**:DISPlay:AXIS:MODE****Command/Query****DESCRIPTION**

The command selects the mode of the axis label.

The query returns the current mode of the axis label.

**COMMAND SYNTAX**

:DISPlay:AXIS:MODE <mode>

<mode>:= {FIXed|MOVing}

- ◆ FIXed means that position of the axes remain fixed, while the coordinates update as the waveform is moving.
- ◆ MOVing means when moving the waveform, the position of the axes moves with the waveform, while the coordinates remain fixed.

**QUERY SYNTAX**

:DISPlay:AXIS:MODE?

**RESPONSE FORMAT**

<mode>

<mode>:= {FIXed|MOVing}

**EXAMPLE**

The following command sets the mode of axis label to FIXed.

Command message:

*:DISPlay:AXIS:MODE FIXed*

*DISP:AXIS:MODE FIXed*

Query message:

*DISP:AXIS:MODE?*

Response message:

*FIXed*

## **:DISPlay:AXIS:POSition**

### **Command/Query**

#### **DESCRIPTION**

The command sets the display position of the vertical axis of the axis label.

The query returns the display position of the vertical axis of the axis label.

#### **COMMAND SYNTAX**

:DISPlay:AXIS:POSition <pos>

<pos>:= {LEFT|MIDDLE|RIGHT}

- ◆ LEFT means the vertical axis is located on the left side of the screen.
- ◆ MIDDLE means the vertical axis is located on the middle side of the screen.
- ◆ RIGHT means the vertical axis is located on the right side of the screen

#### **QUERY SYNTAX**

:DISPlay:AXIS:POSition?

#### **RESPONSE FORMAT**

<pos>

<pos>:= {LEFT|MIDDLE|RIGHT}

#### **EXAMPLE**

The following command sets the vertical axis display position of the axis label to the right.

Command message:

```
:DISPlay:AXIS:POSition RIGHT  
DISP:AXIS:POS RIGH
```

Query message:

```
DISP:AXIS:POS?
```

Response message:

```
RIGHT
```

**:DISPlay:BACKlight****Command/Query****DESCRIPTION**

This command sets the backlight level of the screen.

The query returns the current backlight level of the screen.

**COMMAND SYNTAX**

:DISPlay:BACKlight <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 100]. 0 is the least bright and 100 is the brightest.

**QUERY SYNTAX**

:DISPlay:BACKlight?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the backlight level to 100%.

Command message:

*:DISPlay:BACKlight 100*

*DISP:BACK 100*

Query message:

*DISP:BACK?*

Response message:

*100*

**:DISPlay:CLEar****Command**

**DESCRIPTION** The command clears the waveform displayed on the screen.

**COMMAND SYNTAX** :DISPlay:CLEar

**EXAMPLE** The following command clears the waveform displayed on the screen.

Command message:

*:DISPlay:CLEar*

*DISP:CLE*

**RELATED COMMANDS** :ACQuire:CSWeep

**:DISPlay:COLor****Command/Query**

**DESCRIPTION** The command sets the state of the color grade.

The query returns the state of the current color grade.

**COMMAND SYNTAX** :DISPlay:COLor <state>

<state>:= {ON|OFF}

**QUERY SYNTAX** :DISPlay:COLor?

**RESPONSE FORMAT** <state>

<state>:= {ON|OFF}

**EXAMPLE** The following command turns on the color grade.

Command message:

*:DISPlay:COLor ON*

*DISP:COL ON*

Query message:

*DISP:COL?*

Response message:

*ON*

**:DISPlay:GRATicule****Command/Query****DESCRIPTION**

The command sets the brightness level of the grid.

The query returns the current brightness level of the grid.

**COMMAND SYNTAX**

:DISPlay:GRATicule <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 100]. 0 is the least bright and 100 is the brightest.

**QUERY SYNTAX**

:DISPlay:GRATicule?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the grid brightness level to 50%.

Command message:

*:DISPlay:GRATicule 50*

*DISP:GRAT 50*

Query message:

*DISP:GRAT?*

Response message:

*50*

**:DISPlay:GRIDstyle****Command/Query****DESCRIPTION**

This command selects the type of grid to display.

The query returns the current type of grid to display.

**COMMAND SYNTAX**

:DISPlay:GRIDstyle <type>

<type>:= {FULL|LIGHt|NONE}

**QUERY SYNTAX**

:DISPlay:GRIDstyle?

**RESPONSE FORMAT**

<type>

<type>:= {FULL|LIGHt|NONE}

**EXAMPLE**

The following command sets the grid type to light grid.

Command message:

*:DISPlay:GRIDstyle LIGHt*

*DISP:GRID LIGH*

Query message:

*DISP:GRID?*

Response message:

*LIGHt*

**:DISPlay:HIDemenu****Command/Query****DESCRIPTION**

This command hides the right menu immediately.

**COMMAND SYNTAX**

:DISPlay:HIDemenu

**EXAMPLE**

The following command sets to hide the menu on the right.

Command message:

*:DISPlay:HIDemenu*

*DISP:HID*

**RELATED COMMANDS**

:DISPlay:MENU:HIDE



**:DISPlay:INTensity****Command/Query****DESCRIPTION**

The command sets the intensity level of the waveform.

The query returns the current intensity level of the waveform.

**COMMAND SYNTAX**

:DISPlay:INTensity <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 100]. 0 is the least bright and 100 is the brightest.

**QUERY SYNTAX**

:DISPlay:INTensity?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the intensity level of the waveform to 75%.

Command message:

*:DISPlay:INTensity 75*  
*DISP:INT 75*

Query message:

*DISP:INT?*

Response message:

*75*

**:DISPlay:MENU****Command/Query****DESCRIPTION**

This command selects the style of menu to display.

The query returns the style of menu to display.

**COMMAND SYNTAX**

:DISPlay:MENU <type>

<type>:= {EMBedded|FLOating}

**QUERY SYNTAX**

:DISPlay:MENU?

**RESPONSE FORMAT**

<type>

<type>:= {EMBedded|FLOating}

**EXAMPLE**

The following command sets the menu style to floating.

Command message:

*:DISPlay:MENU FLOating*

*DISP:MENU FLO*

Query message:

*DISP:MENU?*

Response message:

*FLOating*

**:DISPlay:MENU:HIDE**

**Command/Query**

**DESCRIPTION**

This command sets the time for the menu to automatically hide.

The query returns the time for the menu to automatically hide.

**COMMAND SYNTAX**

:DISPlay:MENU:HIDE <time>

<time>:= {OFF|3S|5S|10S|30S|60S}

**QUERY SYNTAX**

:DISPlay:MENU:HIDE?

**RESPONSE FORMAT**

<time>

<time>:= {OFF|3S|5S|10S|30S|60S}

**EXAMPLE**

The following command sets the menu auto hide time to 10s.

Command message:

*:DISPlay:MENU:HIDE 10S*

*DISP:MENU:HIDE 10S*

Query message:

*DISP:MENU:HIDE?*

Response message:

*10S*

**RELATED COMMANDS**

:DISPlay:HIDemenu

**:DISPlay:PERsistence**

**Command/Query**

**DESCRIPTION**

The command selects the persistence duration of the display, in seconds, in persistence mode.

The query returns the current status of the persistence setting.

**COMMAND SYNTAX**

:DISPlay:PERsistence <time>

<time>:= vary from models, see the table below for details.

| Model   | <time>   |
|---|--|
| SDS7000A<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD | {OFF INFinite 100MS 200MS 500MS 1S 5S 10S 30S} |
| SDS2000X Plus<br>SHS800X/SHS1000X<br>SDS1000X HD<br>SDS800X HD                | {OFF INFinite 1S 5S 10S 30S}                   |

**QUERY SYNTAX**

:DISPlay:PERsistence?

**RESPONSE FORMAT**

<time>

**EXAMPLE**

The following command sets the variable persistence at 5 seconds.

Command message:

*:DISPlay:PERsistence 5S*

*DISP:PERs 5S*

Query message:

*DISP:PERs?*

Response message:

*5S*

**:DISPlay:TRANsparencE****Command/Query****DESCRIPTION**

This command sets the transparency level of the information bar, only valid for SHS800X/SHS1000X.

The query returns the transparency level of the current information bar.

**COMMAND SYNTAX**

:DISPlay:TRANsparencE <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 100]. 0 is the least transparent and 100 is the most transparent.

**QUERY SYNTAX**

:DISPlay:TRANsparencE?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the transparency level to 80%.

Command message:

*:DISPlay:TRANsparencE 80*  
*DISP:TRAN 80*

Query message:

*DISP:TRAN?*

Response message:

*80*

**:DISPlay:TYPE****Command/Query****DESCRIPTION**

The command sets the interpolation lines between data points.

The query returns the interpolation lines between data points.

**COMMAND SYNTAX**

:DISPlay:TYPE <type>

<type>:= {VECTor|DOT}

VECTor is the default mode and draws lines between points.

DOT mode displays data more quickly than vector mode but does not draw lines between sample points.

**QUERY SYNTAX**

:DISPlay:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {VECTor|DOT}

**EXAMPLE**

The following command sets the interpolation lines between data points to vector.

Command message:

*:DISPlay:TYPE VECTor*

*DISP:TYPE VECT*

Query message:

*DISP:TYPE?*

Response message:

*VECTor*

## DVM Commands

The :DVM subsystem commands control the digital voltage meter (DVM) feature. This function can be used to measure parameters such as DC and AC amplitudes.

- ◆ **:DVM**
- ◆ **:DVM:ALARm**
- ◆ **:DVM:ARANge**
- ◆ **:DVM:CURREnt**
- ◆ **:DVM:HOLD**
- ◆ **:DVM:MODE**
- ◆ **:DVM:SOURce**

**:DVM****Command/Query****DESCRIPTION**

This command sets the switch of the dvm function.

The query returns the current state of the dvm.

**COMMAND SYNTAX**

:DVM <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DVM?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the dvm.

Command message:

*:DVM ON*

*DVM ON*

Query message:

*DVM?*

Response message:

*ON*



**:DVM:ALARm****Command/Query****DESCRIPTION**

This command sets the switch of the overload alarm. When enabled, an alarm will be given if the signal amplitude exceeds the screen range.

The query returns the switch of the overload arm.

**COMMAND SYNTAX**

:DVM:ALARm <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DVM:ALARm?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}.

**EXAMPLE**

The following command sets the alarm on.

Command message:

*:DVM:ALARm ON*

*DVM:ALAR ON*

Query message:

*DVM:ALAR?*

Response message:

*ON*

**:DVM:ARANge****Command/Query****DESCRIPTION**

This command sets the auto range state for the dvm.

The query returns the auto range state for the dvm.

**COMMAND SYNTAX**

:DVM:ARANge <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DVM:ARANge?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the auto range.

Command message:

*:DVM:ARANge ON*

*DVM:ARAN ON*

Query message:

*DVM:ARAN?*

Response message:

*ON*

**:DVM:CURRent****Query****DESCRIPTION**

The query returns the displayed 3-digit DVM value based on the current mode.

**QUERY SYNTAX**

:DVM:CURRent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following query returns the value of the current dvm mode.

Query message:

*DVM:CURR?*

Response message:

*0.98E+00*

**:DVM:HOLD****Command/Query****DESCRIPTION**

This command sets the hold switch of dvm. When enabled, the measured display value will remain unchanged.

The query returns the current hold switch of dvm.

**COMMAND SYNTAX**

:DVM:HOLD <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:DVM:HOLD?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the hold switch.

Command message:

```
:DVM:HOLD ON  
DVM:HOLD ON
```

Query message:

```
DVM:HOLD?
```

Response message:

```
ON
```

**:DVM:MODE****Command/Query****DESCRIPTION**

This command sets the digital voltmeter (DVM) mode.

The query returns the current digital voltmeter (DVM) mode:.

**COMMAND SYNTAX**

:DVM:MODE <mode>

<mode>:= {DCavg|DCRMs|ACRMs|PKPK|AMPLitude}

- ◆ DCavg displays the DC value of the acquired data.
- ◆ DCRMs displays the root-mean-square value of the acquired data.
- ◆ ACRMs displays the root-mean-square value of the acquired data, with the DC component removed.
- ◆ PKPK displays the difference between maximum and minimum data values
- ◆ AMPLitude displays difference between top and base in a bimodal waveform. If not bimodal, displays difference between max and min

**QUERY SYNTAX**

:DVM:MODE?

**RESPONSE FORMAT**

<mode>

<mode>:= {DCavg|DCRMs|ACRMs|PKPK|AMPLitude}

**EXAMPLE**

The following command sets the dvm mode to AMPLitude.

Command message:

*:DVM:MODE AMPLitude*

*DVM:MODE AMPL*

Query message:

*DVM:MODE?*

Response message:

*AMPLitude*

**:DVM:SOURce****Command/Query****DESCRIPTION**

This command sets the select the analog channel on which digital voltmeter (DVM) measurements are made.

The query returns the current source of dvm.

**COMMAND SYNTAX**

:DVM:SOURce <source>

<source>:= {C<n>}

- ◆ C denotes an analog channel.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:DVM:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {Cn}

**EXAMPLE**

The following command sets the dvm source to C2.

Command message:

*:DVM:SOURce C2*

*DVM:SOUR C2*

Query message:

*DVM:SOUR?*

Response message:

*C2*

## **FUNCTION Commands**

The :FUNCTION subsystem commands control the math functions in the oscilloscope.

- ◆ **:FUNCTION:FFTDISplay**
- ◆ **:FUNCTION:GVALue**
- ◆ **:FUNCTION<x>**
- ◆ **:FUNCTION<x>:AVERAge:NUM**
- ◆ **:FUNCTION<x>:DIFF:DX**
- ◆ **:FUNCTION<x>:ERES:BITS**
- ◆ **:FUNCTION<x>:FFT:AUTOset**
- ◆ **:FUNCTION<x>:FFT:HCENter**
- ◆ **:FUNCTION<x>:FFT:HSCale**
- ◆ **:FUNCTION<x>:FFT:SPAN**
- ◆ **:FUNCTION<x>:FFT:LOAD**
- ◆ **:FUNCTION<x>:FFT:MODE**
- ◆ **:FUNCTION<x>:FFT:POINts**
- ◆ **:FUNCTION<x>:FFT:RESET**
- ◆ **:FUNCTION<x>:FFT:RLEVel**
- ◆ **:FUNCTION<x>:FFT:SCALe**
- ◆ **:FUNCTION<x>:FFT:SEARCh**
- ◆ **:FUNCTION<x>:FFT:SEARCh:EXCURsion**
- ◆ **:FUNCTION<x>:FFT:SEARCh:MARKer<n>**
- ◆ **:FUNCTION<x>:FFT:SEARCh:MARKer<n>:SHOW**
- ◆ **:FUNCTION<x>:FFT:SEARCh:MON**
- ◆ **:FUNCTION<x>:FFT:SEARCh:PORDer**
- ◆ **:FUNCTION<x>:FFT:SEARCh:RESult**
- ◆ **:FUNCTION<x>:FFT:SEARCh:TABLE**
- ◆ **:FUNCTION<x>:FFT:SEARCh:TABLE:DELTA**
- ◆ **:FUNCTION<x>:FFT:SEARCh:TABLE:FREQuency**
- ◆ **:FUNCTION<x>:FFT:SEARCh:THReshold**
- ◆ **:FUNCTION<x>:FFT:UNIT**
- ◆ **:FUNCTION<x>:FFT:WINDow**
- ◆ **:FUNCTION<x>:FILTer:TYPE**
- ◆ **:FUNCTION<x>:FILTer:HFRequency**
- ◆ **:FUNCTION<x>:FILTer:LFRequency**
- ◆ **:FUNCTION<x>:INTegrate:GATE**
- ◆ **:FUNCTION<x>:INTERpolate:COEF**

- ◆ **:FUNction<x>:INVert**
- ◆ **:FUNction<x>:LABel**
- ◆ **:FUNction<x>:LABel:TEXT**
- ◆ **:FUNction<x>:MAXHold:SWeeps**
- ◆ **:FUNction<x>:MINHold:SWeeps**
- ◆ **:FUNction<x>:OPERation**
- ◆ **:FUNction<x>:POSition**
- ◆ **:FUNction<x>:SCALe**
- ◆ **:FUNction<x>:SOURce1**
- ◆ **:FUNction<x>:SOURce2**



**:FUNction:FFTDisplay****Command/Query****DESCRIPTION**

This command sets the display mode of the FFT waveform.

This query returns the current display mode of the FFT waveform.

**COMMAND SYNTAX**

:FUNction:FFTDisplay <mode>

<mode>:= {SPLit|FULL|EXCLusive}

- ◆ SPLit means that the channel waveform and the FFT waveform are displayed on the screen separately.
- ◆ FULL means a full-screen display of the FFT waveform.
- ◆ EXCLusive means that only the FFT waveform is displayed on the screen.

**QUERY SYNTAX**

:FUNction:FFTDisplay?

**RESPONSE FORMAT**

<mode>

<mode>:= {SPLit|FULL|EXCLusive}

**EXAMPLE**

The following command sets the display mode of the FFT waveform to split.

Command message:

```
:FUNction:FFTDisplay SPLit  
FUNC:FFTD SPL
```

Query message:

```
FUNC:FFTD?
```

Response message:

```
SPLit
```

**:FUNction:GVALue****Command/Query****DESCRIPTION**

The command sets the integration threshold value of gate A and gate B.

The query returns the current integration threshold values.

**COMMAND SYNTAX**

:FUNction:GVALue <valueA>,<valueB>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-horizontal\_grid/2\*timebase+delay, horizontal\_grid/2\*timebase+delay].

**Note:**

The value of GA cannot be greater than that of GB. If you set the value greater than GB, it will automatically be set to the same value as GB.

**QUERY SYNTAX**

:FUNction:GVALue?

**RESPONSE FORMAT**

<valueA>,<valueB>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the position of gate A to -100 ns and set the position of gate B to 100ns.

Command message:

```
:FUNction:GVALue -1.00E-07,1.00E-07
FUNC:GVAL -1.00E-07,1.00E-07
```

Query message:

```
FUNC:GVAL?
```

Response message:

```
-1.00E-07,1.00E-07
```

**RELATED COMMANDS**

:FUNction<x>:INTegrate:GATE

**:FUNction<x>****Command/Query****DESCRIPTION**

This command set the switch of the math function.

This query returns the current state of the math function.

**COMMAND SYNTAX**

:FUNction<x> <state>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNction<x>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables Function1 of math.

Command message:

*:FUNction1 ON*

*FUNC1 ON*

Query message:

*FUNC1?*

Response message:

*ON*

**:FUNction<x>:AVERage:NUM**

**Command/Query**

**DESCRIPTION**

This command sets the average number for the average operation.

This query returns the current average number for the average operation.

**COMMAND SYNTAX**

:FUNction<x>:AVERage:NUM <num>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<num>:= vary from models, see the table below for details.

| Model   | <num>  |
|---|--|
| SDS7000A<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD | {4 16 32 64 128 256 512 1024 2048 4096 8192} |
| SDS2000X Plus<br>SDS1000X HD<br>SDS800X HD                                    | {4 16 32 64 128 256 512 1024}                |

**QUERY SYNTAX**

:FUNction<x>:AVERage:NUM?

**RESPONSE FORMAT**

<num>

**EXAMPLE**

The following command changes the average number for the average operation to 128 on Function2.

Command message:

*:FUNction2:AVERage:NUM 128*  
*FUNC2:AVER:NUM 128*

Query message:

*FUNC2:AVER:NUM?*

Response message:

*128*

**:FUNction<x>:DIFF:DX****Command/Query****DESCRIPTION**

This command sets the step size of the differential operation.

This query returns the current step size of the differential operation.

**COMMAND SYNTAX**

:FUNction<x>:DIFF:DX <dx>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command

<dx>:= Value in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:FUNction<x>:DIFF:DX?

**RESPONSE FORMAT**

<dx>

<dx>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the step of the differential operation to 4 on Function1.

Command message:

*:FUNction1:DIFF:DX 4*

*FUNC1:DIFF:DX 4*

Query message:

*FUNC1:DIFF:DX?*

Response message:

*4*

**:FUNcTion<x>:ERES:BITS****Command/Query****DESCRIPTION**

This command sets the eres bits for the eres operation.

This query returns the current eres bits for the eres operation.

**COMMAND SYNTAX**

:FUNcTion<n>:ERES:BITS <bits>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<bits>:= {0.5|1.0|1.5|2.0|2.5|3.0}

**QUERY SYNTAX**

:FUNcTion<x>:ERES:BITS?

**RESPONSE FORMAT**

<bits>:= {0.5|1.0|1.5|2.0|2.5|3.0}

**EXAMPLE**

The following command changes the eres bits for the eres operation to 3.0 on Function2.

Command message:

*:FUNcTion2:ERES:BITS 3.0*

*FUNC2:ERES:BITS 3.0*

Query message:

*FUNC2:ERES:BITS?*

Response message:

*3.0*

**:FUNcTion<x>:FFT:AUToset****Command****DESCRIPTION**

This command causes the FFT waveform to be displayed at the best position on the screen.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:AUToset <mode>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to on FUNcTion and defines the math that is affected by the command.

<mode>:= {SPAN|PEAK|NORMal}

- ◆ SPAN – full span.
- ◆ PEAK – center to peak.
- ◆ NORMal –center set to the fundamental frequency and the span is set to one-half of the fft sampling rate

**EXAMPLE**

The following command causes the FFT waveform to be displayed at the best position on the screen on Function2.

Command message:

```
:FUNcTion2:FFT:AUToset NORMal  
FUNC2:FFT:AUT NORM
```

**:FUNcTion<x>:FFT:HcENter****Command/Query****DESCRIPTION**

This command sets the center frequency of FFT.

This query returns the current center frequency of FFT.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:HcENter <center>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<center>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The range of legal values varies with the FFT sampling rate.

**QUERY SYNTAX**

:FUNcTion<x>:FFT:HcENter?

**RESPONSE FORMAT**

<center>

<center>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the center frequency of FFT to 2 MHz on Function2.

Command message:

*:FUNcTion2:FFT:HcENter 2.00E+06*

*FUNC2:FFT:HCEN 2.00E+06*

Query message:

*FUNC2:FFT:HCEN?*

Response message:

*2.00E+06*



**:FUNCTION<x>:FFT:HSCale****Query****DESCRIPTION**

This query returns the current horizontal scale of FFT.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:HSCale?

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

**RESPONSE FORMAT**

<scale>

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following query returns the horizontal scale of FFT on Function2.

Query message:

*FUNC2:FFT:HSC?*

Response message:

*1.00E+08*

**RELATED COMMANDS**

:FUNCTION<x>:FFT:HCENter

:FUNCTION<x>:FFT:SPAN

**:FUNcTion<x>:FFT:SPAN****Command/Query****DESCRIPTION**

This command sets the horizontal span of FFT.

This query returns the current horizontal span of FFT.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:SPAN <span>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<span>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:FUNcTion<x>:FFT:SPAN?

**RESPONSE FORMAT**

<span>

<span>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the span frequency of FFT to 2 MHz on Function2.

Command message:

```
:FUNcTion2:FFT:SPAN 2.00E+06
FUNC2:FFT:SPAN 2.00E+06
```

Query message:

```
FUNC2:FFT:SPAN?
```

Response message:

```
1.00E+08
```

**RELATED COMMANDS**

:FUNcTion<x>:FFT:HcENter

**:FUNCTION<x>:FFT:LOAD****Command/Query****DESCRIPTION**

This command sets the external load of the FFT.

This query returns the current external load of FFT.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:LOAD <load>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<load>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 1000000]

**Note:**

The load can be set only when the FFT unit is dBm.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:LOAD?

**RESPONSE FORMAT**

<load>

<load>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the load of FFT to 50Ω on Function2.

Command message:

*:FUNCTION2:FFT:LOAD 50*

*FUNC2:FFT:LOAD 50*

Query message:

*FUNC2:FFT:LOAD?*

Response message:

*50*

**RELATED COMMANDS**

:FUNCTION<x>:FFT:UNIT

**:FUNcTion<x>:FFT:MODE****Command/Query****DESCRIPTION**

This command selects the acquisition mode of the FFT operation.

This query returns the current acquisition mode of the FFT operation.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:MODE <mode>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<mode>:= {NORMal|MAXHold|AVERAge[,<num>]}

- ◆ NORMal sets the FFT in the normal mode.
- ◆ MAXHold sets the FFT in the max detect mode.
- ◆ AVERAge sets the FFT in the averaging mode.

<num>:= Value in NR1 format, including an integer and no decimal point, like 1.

The range of the value is [4, 1024].

**QUERY SYNTAX**

:FUNcTion<x>:FFT:MODE?

**RESPONSE FORMAT**

<mode>

<mode>:= {NORMal|MAXHold|AVERAge[,<num>]}

<num>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the acquisition mode of the FFT operation on Function2 to normal.

Command message:

```
:FUNcTion2:FFT:MODE NORMal
FUNC2:FFT:MODE NORM
```

Query message:

```
FUNC2:FFT:MODE?
```

Response message:

```
NORMal
```

**:FUNCTION<x>:FFT:POINTS**

**Command/Query**

**DESCRIPTION**

This command sets the maximum number of points for the FFT operation.

This query returns the current maximum number of points for the FFT operation.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:POINTS <point>

<n>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<point>:= Vary from models, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | {1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M 4M 8M 16M 32M } |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                   | {1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M 4M 8M}          |
| SDS3000X HD   | {1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M 4M}             |
| SDS5000X<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | {1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M}                |
| SHS800X<br>SHS1000X   | {1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M}                   |

**QUERY SYNTAX**

:FUNCTION<x>:FFT:POINTS?

**RESPONSE FORMAT**

<point>

**EXAMPLE**

The following command changes the maximum number of points for the FFT operation to 2M on Function2.

Command message:

*:FUNCTION2:FFT:POINTS 2M*  
*FUNC2:FFT:POIN 2M*

Query message:

*FUNC2:FFT:POIN?*

Response message:

*2M*

**:FUNCTION<x>:FFT:RESET****Command****DESCRIPTION**

This command restarts counting when the acquisition mode is average.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:RESET

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

**EXAMPLE**

The following command restarts counting on Function2 when the acquisition mode is average.

Command message:

*:FUNCTION2:FFT:RESET*

*FUNC2:FFT:RESET*

**RELATED COMMANDS**

:FUNCTION<x>:FFT:MODE

**:FUNCTION<x>:FFT:RLEVEL****Command/Query****DESCRIPTION**

The command sets the reference level of the FFT operation.

The query returns the current reference level of the FFT operation.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:RLEVEL <level>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<level>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the values is related to the probe of the FFT source.

| Probe  | dBVrms     | Vrms         | dBm        |
|--------|------------|--------------|------------|
| 1E6 X  | [-40,200]  | [1E-2,1E10]  | [-27,213]  |
| 1E5 X  | [-60,180]  | [1E-3,1E9]   | [-47,193]  |
| 1E4 X  | [-80,160]  | [1E-4,1E8]   | [-67,173]  |
| 1000X  | [-100,140] | [1E-5,1E7]   | [-87,153]  |
| 100X   | [-120,120] | [1E-6,1E6]   | [-107,133] |
| 10X    | [-140,100] | [1E-7,1E5]   | [-127,113] |
| 1      | [-160,80]  | [1E-8,1E4]   | [-147,93]  |
| 0.1X   | [-180,60]  | [1E-9,1E3]   | [-167,73]  |
| 0.01X  | [-200,40]  | [1E-10,1E2]  | [-187,53]  |
| 1E-3 X | [-220,20]  | [1E-11,10]   | [-207,33]  |
| 1E-4 X | [-240,0]   | [1E-12,1]    | [-227,13]  |
| 1E-5 X | [-260,-20] | [1E-13,1E-1] | [-247,-7]  |
| 1E-6 X | [-280,-40] | [1E-14,1E-2] | [-267,-27] |

**Note:**

The smaller the :FUNCTION<x>:FFT:SCALE, the greater the accuracy of the level value.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:RLEVEL?

**RESPONSE FORMAT**

<level>

<level>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the reference level of FFT operation to 10 dBV on Function2 when the FFT unit is dBVrms.

Command message:

```
:FUNction2:FFT:RLEVel 1.00E+01  
FUNC2:FFT:RLEV 1.00E+01
```

Query message:

```
FUNC2:FFT:RLEV?
```

Response message:

```
1.00E+01
```

**RELATED COMMANDS**

```
:CHANnel<n>:PROBe  
:FUNction<x>:FFT:SCALe
```



**:FUNction<x>:FFT:SCALe**

**Command/Query**

**DESCRIPTION**

The command sets the vertical scale of the FFT.

The query returns the current vertical scale of FFT.

**COMMAND SYNTAX**

:FUNction<x>:FFT:SCALe <scale>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the values is related to the vertical unit.

| Unit   | Range                |
|--------|----------------------|
| dBVrms | [1.00E-01, 2.00E+01] |
| Vrms   | [1.00E-03, 1.00E+01] |
| dBm    | [1.00E-01, 2.00E+01] |

**QUERY SYNTAX**

:FUNction<x>:FFT:SCALe?

**RESPONSE FORMAT**

<scale>

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the vertical scale of FFT to 20 dB on Function2 when the FFT unit is dBVrms.

Command message:

```
:FUNction2:FFT:SCALe 2.00E+01
FUNC2:FFT:SCAL 2.00E+01
```

Query message:

```
FUNC2:FFT:SCAL?
```

Response message:

```
2.00E+01
```

**RELATED COMMANDS**

:CHANnel<n>:PROBE

**:FUNction<x>:FFT:SEARch****Command/Query****DESCRIPTION**

This command selects the search tools type of the FFT operation.

This query returns the current search tools type of the FFT operation.

**COMMAND SYNTAX**

:FUNction<x>:FFT:SEARch <type>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<type>:= {OFF|PEAK|MARKer}

**QUERY SYNTAX**

:FUNction<x>:FFT:SEARch?

**RESPONSE FORMAT**

<type>

<type>:= {OFF|PEAK|MARKer}

**EXAMPLE**

The following command sets the search tools type of FFT operation on Function2 to marker.

Command message:

```
:FUNction2:FFT:SEARch MARKer
FUNC2:FFT:SEAR MARK
```

Query message:

```
FUNC2:FFT:SEAR?
```

Response message:

```
MARKer
```

**RELATED COMMANDS**

```
:FUNction<x>:FFT:SEARch:THReshold
:FUNction<x>:FFT:SEARch:EXCursion
```

**:FUNCTION<x>:FFT:SEARCh:EXCURSION****Command/Query****DESCRIPTION**

This command sets the search excursion of the search tool (marker or peak) for the FFT operation.

This query returns the current search excursion of the search tool for the FFT operation.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARCh:EXCURSION <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the values is [0, 1.60E+02] when the FFT unit is dBVrms. The value range varies with the corresponding unit.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:EXCURSION?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the search excursion of the marker of the FFT operation to 20 dB on Function2 when the FFT unit is dBVrms.

Command message:

```
:FUNCTION2:FFT:SEARCh:EXCURSION 2.00E+01  
FUNC2:FFT:SEAR:EXC 2.00E+01
```

Query message:

```
FUNC2:FFT:SEAR:EXC?
```

Response message:

```
2.00E+01
```

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCh:THRShold

**:FUNction<x>:FFT:SEARch:MARKer<n>****Command****DESCRIPTION**

This command sets the frequency of the specified marker.

**COMMAND SYNTAX**

:FUNction<x>:FFT:SEARch:MARKer<n> <freq>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to on FUNction and defines the math that is affected by the command.

<n>:= 1 to (# marker index) in NR1 format, is attached as a suffix to on MARKer and defines the marker that is affected by the command.

<freq>:= {NPeak|NAMPlitude|<value>}

- ◆ NPeak means marking the next peak
- ◆ NAMPlitude means marking the next amplitude
- ◆ <value>:= frequency value in NR3 format.

**EXAMPLE**

The following command sets the frequency of FFT marker1 for function1 to 1 MHz.

Command message:

```
:FUNction1:FFT:SEARch:MARKer1 1.00E+06
```

```
FUNC1:FFT:SEAR:MARK1 1.00E+06
```

**RELATED COMMANDS**

```
:FUNction<x>:FFT:SEARch
```

```
:FUNction<x>:FFT:SEARch:MARKer<n>:SHOW
```

**:FUNcTion<x>:FFT:SEARch:MARKer<n>:SHOW****Command****DESCRIPTION**

This command sets the switch state of the specified marker.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:SEARch:MARKer<n>:SHOW <state>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to on FUNcTion and defines the math that is affected by the command.

<n>:= 1 to (# marker index) in NR1 format, is attached as a suffix to on MARKer and defines the marker that is affected by the command.

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets to disable function1 FFT search marker2

Command message:

*:FUNcTion1:FFT:SEARch:MARKer2:SHOW ON*  
*FUNC1:FFT:SEAR:MARK2:SHOW ON*

**RELATED COMMANDS**

:FUNcTion<x>:FFT:SEARch

**:FUNcTion<x>:FFT:SEARch:MON****Command****DESCRIPTION**

This command sets the FFT marker on peaks or harmonics.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:SEARch:MON <type>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to on FUNcTion and defines the math that is affected by the command.

<type>:= {PEAK|HARMonics}

- ◆ PEAK means marking on peaks
- ◆ HARMonics means marking on harmonics

**EXAMPLE**

The following command sets the marker of function 1 FFT on PEAK.

Command message:

*:FUNcTion1:FFT:SEARch:MON PEAK*

*FUNC1:FFT:SEAR:MON PEAK*

**:FUNCTION<x>:FFT:SEARCh:PORDer****Command/Query****DESCRIPTION**

This command sets the sorting method for FFT peak values.

The query returns the sorting method for FFT peak values.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARCh:PORDer <type>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to on FUNCTION and defines the math that is affected by the command.

<type>:= {AMPLitude|FREQuency}

- ◆ AMPLitude means sorting by amplitude.
- ◆ FREQuency means sorting by frequency.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:PORDer?

**RESPONSE FORMAT**

<type>

<type>:= {AMPLitude|FREQuency}

**EXAMPLE**

The following command sets the FFT list of function1 in order of amplitude.

Command message:

```
:FUNCTION1:FFT:SEARCh:PORDer AMPLitude  
FUNC1:FFT:SEAR:POrD AMPL
```

Query message:

```
FUNC1:FFT:SEAR:POrD?
```

Response message:

```
AMPLitude
```

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCh

**:FUNCTION<x>:FFT:SEARCh:RESult**

**Query**

**DESCRIPTION**

The query returns the current search list result for the FFT operation. It only contains search number, frequency and amplitude information.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:RESult?

**RESPONSE FORMAT**

<type>,<no>,<freq>,<ampl>;

<type>:={Markers|Peaks}

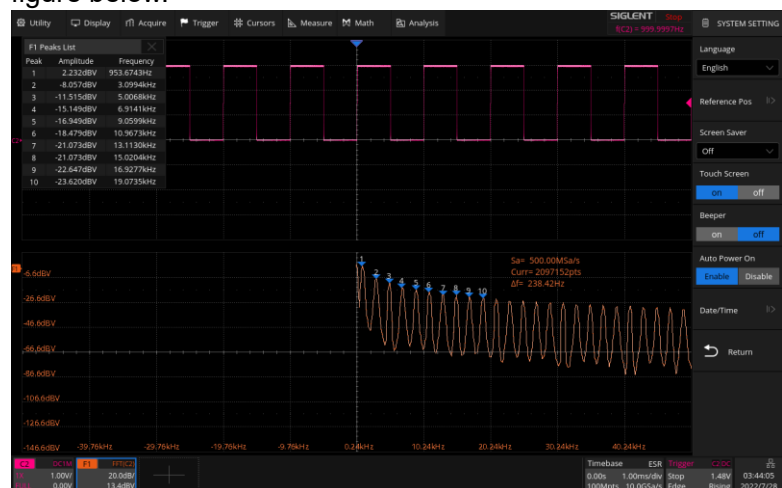
<no>:= Value in NR1 format, indicates the peak number or marker number

<freq>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

<ampl>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The unit is the same as FFT vertical unit

**EXAMPLE**

The following query returns the peaks result of function1 in the figure below.



Query message:

*FUNC1:FFT:SEAR:RES?*

Response message:

*Peaks,1,9.536743E+02,2.231755E+00;  
 2,3.099442E+03,-8.056905E+00;  
 3,5.006790E+03,-1.151463E+01;  
 4,6.914139E+03,-1.514894E+01;  
 5,9.059906E+03,-1.694874E+01;  
 6,1.096725E+04,-1.847880E+01;  
 7,1.311302E+04,-2.107302E+01;  
 8,1.502037E+04,-2.107302E+01;  
 9,1.692772E+04,-2.264706E+01;  
 10,1.907349E+04,-2.361992E+01;*



**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCH:THRESHOLD  
:FUNCTION<x>:FFT:SEARCH:EXCURSION  
:FUNCTION<x>:FFT:UNIT

**:FUNCTION<x>:FFT:SEARCh:TABLE****Command/Query****DESCRIPTION**

This command sets the state of FFT search table.

The query returns the state of FFT search table.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARCh:TABLE <state>

<x>:= 1 to (# math functions) in NR1 format is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:TABLE?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the search table state of function1 FFT to ON.

Command message:

```
:FUNCTION1:FFT:SEARCh:TABLE ON
FUNC1:FFT:SEAR:TABL ON
```

Query message:

```
FUNC1:FFT:SEAR:TABL?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCh

**:FUNCTION<x>:FFT:SEARCh:TABLE:DELTA****Command/Query****DESCRIPTION**

This command sets the display state of increments in the FFT search table.

The query returns the the display state of increments in the FFT search table.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARCh:TABLE:DELTA <state>

<x>:= 1 to (# math functions) in NR1 format is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:TABLE:DELTA?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the function 1 FFT's search table to display increment.

Command message:

*:FUNCTION1:FFT:SEARCh:TABLE:DELTA ON*  
*FUNC1:FFT:SEAR:TABL:DELT ON*

Query message:

*FUNC1:FFT:SEAR:TABL:DELT?*

Response message:

*ON*

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCh:TABLE

**:FUNCTION<x>:FFT:SEARch:TABLE:FREQuency****Command/Query****DESCRIPTION**

This command sets the display state of frequency in the FFT search table.

This query returns the the display state of frequency in the FFT search table.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARch:TABLE:FREQuency <state>

<x>:= 1 to (# math functions) in NR1 format is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARch:TABLE:FREQuency?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the function 1 FFT's search table to display frequency.

Command message:

```
:FUNCTION1:FFT:SEARch:TABLE:FREQuency ON
FUNC1:FFT:SEAR:TABL:FREQ ON
```

Query message:

```
FUNC1:FFT:SEAR:TABL:FREQ?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARch:TABLE

**:FUNCTION<x>:FFT:SEARCh:THReshold****Command/Query****DESCRIPTION**

The command sets the search threshold of the search tool (marker or peak) for the FFT operation.

The query returns the current search threshold of the search tool for the FFT operation.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:SEARCh:THReshold <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the values is [-1.60E+02, 8.00E+01], when FFT unit is dBVrms. The value changes to match the set Units value.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:SEARCh:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the search threshold of the marker of the FFT operation to -100 dBV on Function2 when the FFT unit is dBVrms.

Command message:

```
:FUNCTION2:FFT:SEARCh:THReshold -1.00E+2  
FUNC2:FFT:SEAR:THR -1.00E+2
```

Query message:

```
FUNC2:FFT:SEAR:THR?
```

Response message:

```
-1.00E+02
```

**RELATED COMMANDS**

:FUNCTION<x>:FFT:SEARCh:EXCursion

**:FUNcTion<x>:FFT:UNIT****Command/Query****DESCRIPTION**

This command sets the unit type of the FFT operation.

This query returns the current unit type of the FFT operation.

**COMMAND SYNTAX**

:FUNcTion<x>:FFT:UNIT <unit>

<x>:= 1 to (# math functions) in NR1 format is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<unit>:= {DBVrms|Vrms|DBm}

**QUERY SYNTAX**

:FUNcTion<x>:FFT:UNIT?

**RESPONSE FORMAT**

<unit>

<unit>:= {DBVrms|Vrms|DBm}

**EXAMPLE**

The following command sets the unit type of FFT operation on Function2 to dBVrms.

Command message:

*:FUNcTion2:FFT:UNIT DBVrms*  
*FUNC2:FFT:UNIT DBVrms*

Query message:

*FUNC2:FFT:UNIT?*

Response message:

*DBVrms*

**:FUNCTION<x>:FFT:WINDow****Command/Query****DESCRIPTION**

This command selects the window type of the FFT operation.

This query returns the current window type of the FFT operation.

**COMMAND SYNTAX**

:FUNCTION<x>:FFT:WINDow <window>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<window>:=

{RECTangle|BLACKman|HANNing|HAMMING|FLATtop}

- ◆ RECTangle is useful for transient signals, and signals where there are an integral number of cycles in the time record.
- ◆ BLACKman reduces time resolution compared to the rectangular window, but it improves the capacity to detect smaller impulses due to lower secondary lobes (provides minimal spectral leakage).
- ◆ HANNing 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.
- ◆ HAMMING means Hamming.
- ◆ FLATtop is the best for making accurate amplitude measurements of frequency peaks.

**QUERY SYNTAX**

:FUNCTION<x>:FFT:WINDow?

**RESPONSE FORMAT**

<window>

<window>:=

{RECTangle|BLACKman|HANNing|HAMMING|FLATtop}

**EXAMPLE**

The following command sets the windowing of the FFT operation on Function2 to Flattop.

Command message:

```
:FUNCTION2:FFT:WINDow FLATtop
FUNC2:FFT:WIND FLAT
```

Query message:

```
FUNC2:FFT:WIND?
```

Response message:

```
FLATtop
```

**:FUNcTion<x>:FILTer:TYPe****Command/Query****DESCRIPTION**

This command selects the filter type of the filter operation.

This query returns the current filter type of the filter operation.

**COMMAND SYNTAX**

:FUNcTion<x>:FILTer:TYPe <type>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<type>:= {LPASs|HPASs|BPASs|BREJect}

- ◆ LPASs - Low pass filter.
- ◆ HPASs - High pass filter.
- ◆ BPASs - Band pass filter.
- ◆ BREJect - Band reject filter.

**QUERY SYNTAX**

:FUNcTion<x>:FILTer:TYPe?

**RESPONSE FORMAT**

<type>

<type>:= {LPASs|HPASs|BPASs|BREJect}

**EXAMPLE**

The following command sets the filter type of the filter operation on Function2 to HPASs.

Command message:

```
:FUNcTion2:FILTer:TYPe HPASs
FUNC2:FILT:TYP HPAS
```

Query message:

```
FUNC2:FILT:TYP?
```

Response message:

```
HPASs
```



**:FUNCTION<x>:FILTER:HFRrequency****Command/Query****DESCRIPTION**

This command sets the upper frequency of the filter.

This query returns the current upper frequency of the filter.

The command/query is available only when the filter type is BPASs or BREject.

**COMMAND SYNTAX**

:FUNCTION<x>:FILTER:HFRrequency <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:FUNCTION<x>:FILTER:HFRrequency?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper freq for the filter operation to 100MHz on Function2.

Command message:

*:FUNCTION2:FILTER:HFRrequency 100MHz*

*FUNC2:FILT:HFR 100MHz*

Query message:

*FUNC2:FILT:HFR?*

Response message:

*1.00E+08*

**RELATED COMMANDS**

:FUNCTION<x>:FILTER:TYPE

:FUNCTION<x>:FILTER:LFRrequency

**:FUNCTION<x>:FILTER:LFRrequency****Command/Query****DESCRIPTION**

This command sets the lower frequency of the filter.

This query returns the current lower frequency of the filter.

**COMMAND SYNTAX**

:FUNCTION<x>:FILTER:LFRrequency <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:FUNCTION<x>:FILTER:LFRrequency?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower freq for the filter operation to 50MHz on Function2.

Command message:

```
:FUNCTION2:FILTER:LFRrequency 50MHz
FUNC2:FILT:LFR 50MHz
```

Query message:

```
FUNC2:FILT:LFR?
```

Response message:

```
5.00E+07
```

**RELATED COMMANDS**

:FUNCTION<x>:FILTER:HFRrequency

**:FUNCTION<x>:INTEGRATE:GATE****Command/Query****DESCRIPTION**

This command selects whether to enable the threshold of the integral operation.

This query returns the threshold status of the integral operation.

**COMMAND SYNTAX**

:FUNCTION<x>:INTEGRATE:GATE <state>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNCTION<x>:INTEGRATE:GATE?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the threshold for the integral operation of function 1.

Command message:

*:FUNCTION1:INTEGRATE:GATE ON*  
*FUNC1:INT:GATE ON*

Query message:

*FUNC1:INT:GATE?*

Response message:

*ON*

**RELATED COMMANDS**

:FUNCTION:GVALue

**:FUNction<x>:INtegrate:OFFSet****Command/Query****DESCRIPTION**

The command sets the dc offset of the integrate operation.

The query returns the current dc offset of the integrate operation.

**COMMAND SYNTAX**

:FUNction<x>:INtegrate:OFFSet <offset>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of values corresponds to the vertical pixel range: [-100 pixels, 100 pixels].

**QUERY SYNTAX**

:FUNction<x>:INtegrate:OFFSet?

**RESPONSE FORMAT**

<offset>

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command changes the offset of the integral operation to 100 mV on Function1.

Command message:

```
:FUNction1:INtegrate:OFFSet 1.00E-01
FUNC1:INT:OFFS 1.00E-01
```

Query message:

```
FUNC1:INT:OFFS?
```

Response message:

```
1.00E-01
```

**RELATED COMMANDS**

:CHANnel<n>:PROBE

**:FUNCTION<x>:INTERpolate:COEF****Command/Query****DESCRIPTION**

This command sets the upsample coef for the interpolate operation.

This query returns the current upsample coef for the interpolate operation.

**COMMAND SYNTAX**

:FUNCTION<x>:INTERpolate:COEF <coef>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<coef>:= {2|5|10|20}

**QUERY SYNTAX**

:FUNCTION<x>:INTERpolate:COEF?

**RESPONSE FORMAT**

<coef>:= {2|5|10|20}

**EXAMPLE**

The following command changes the upsample coef for the interpolate operation to 10 on Function2.

Command message:

*:FUNCTION2:INTERpolate:COEF 10*  
*FUNC2:INTE:COEF 10*

Query message:

*FUNC2:INTE:COEF?*

Response message:

*10*

**:FUNcTion<x>:INVert****Command/Query****DESCRIPTION**

This command inverts the math waveform.

This query returns whether the math waveform is inverted or not.

**COMMAND SYNTAX**

:FUNcTion<x>:INVert <state>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNcTion<x>:INVert?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command inverts the Function1 waveform.

Command message:

*:FUNcTion1:INVert ON*  
*FUNC1:INV ON*

Query message:

*FUNC1:INV?*

Response message:

*ON*

**:FUNcTion<x>:LABel****Command/Query****DESCRIPTION**

This command is to turn the specified math label on or off.

This query returns the label associated with a particular math function.

**COMMAND SYNTAX**

:FUNcTion<x>:LABel <state>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:FUNcTion<x>:LABel?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the label of the Function1.

Command message:  
*:FUNcTion1:LABel ON*  
*FUNC1:LAB ON*

Query message:  
*FUNC1:LAB?*

Response message:  
*ON*

**RELATED COMMANDS**

:FUNcTion<x>:LABel:TEXT

**:FUNcTion<x>:LABel:TEXT****Command/Query****DESCRIPTION**

This command sets the selected math label to the string that follows. Setting a label for a math function also adds the name to the label list in non-volatile memory (replacing the oldest label in the list)

This query returns the current label text of the selected math.

**COMMAND SYNTAX**

:FUNcTion<x>:LABel:TEXT <string>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:FUNcTion<x>:LABel:TEXT?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the label text of the Function1 to "MATH".

Command message:

*:FUNcTion1:LABel:TEXT "MATH"*

*FUNC1:LAB:TEXT "MATH"*

Query message:

*FUNC1:LAB:TEXT?*

Response message:

*"MATH"*

**RELATED COMMANDS**

:FUNcTion<x>:LABel



**:FUNCTION<x>:MAXHold:SWeeps****Command/Query****DESCRIPTION**

This command sets the sweeps limit for the maxhold operation.

This query returns the current sweeps limit for the maxhold operation.

**COMMAND SYNTAX**

:FUNCTION<x>:MAXHold:SWeeps <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 2147483646], where 0 represents unlimited.

**QUERY SYNTAX**

:FUNCTION<x>:MAXHold:SWeeps?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the sweeps limit for the maxhold operation to 100 on Function2.

Command message:

```
:FUNCTION2:MAXHold:SWeeps 100  
FUNC2:MAXH:SW 100
```

Query message:

```
FUNC2:MAXH:SW?
```

Response message:

```
100
```

**:FUNcTion<x>:MINHold:SWeeps****Command/Query****DESCRIPTION**

This command sets the sweeps limit for the minhold operation.

This query returns the current sweeps limit for the minhold operation.

**COMMAND SYNTAX**

:FUNcTion<x>:MINHold:SWeeps <value>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNcTion and defines the math that is affected by the command.

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 2147483646], where 0 represents unlimited.

**QUERY SYNTAX**

:FUNcTion<x>:MINHold:SWeeps?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the sweeps limit for the minhold operation to 100 on Function2.

Command message:

```
:FUNcTion2:MINHold:SWeeps 100  
FUNC2:MINH:SW 100
```

Query message:

```
FUNC2:MINH:SW?
```

Response message:

```
100
```

**:FUNCTION<x>:OPERation**

**Command/Query**

**DESCRIPTION**

This command sets the desired waveform math operation.

This query returns the current operation for the selected function.

**COMMAND SYNTAX**

:FUNCTION<x>:OPERation <operation>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<operation>:=  
 {ADD|SUBTRACT|MULTIPLY|DIVISION|INTEGRATE|DIFF|FFT|SQRT|  
 ERES|AVERAGE|ABSOLUTE|SIGN|IDENTITY|NEGATION|EXP|TEN|  
 LN|LOG|INTERPOLATE|MAXHOLD|MINHOLD|FILTER}

**QUERY SYNTAX**

:FUNCTION<x>:OPERation?

**RESPONSE FORMAT**

<operation>

<operation>:=  
 {ADD|SUBTRACT|MULTIPLY|DIVISION|INTEGRATE|DIFF|FFT|SQRT|  
 ERES|AVERAGE|ABSOLUTE|SIGN|IDENTITY|NEGATION|EXP|TEN|  
 LN|LOG|INTERPOLATE|MAXHOLD|MINHOLD|FILTER}

**EXAMPLE**

The following command sets the Function1 operation to Multiplication.

Command message:  
*:FUNCTION1:OPERation MULTIpLy*  
*FUNC1:OPER MULT*

Query message:  
*FUNC1:OPER?*

Response message:  
*MULTIpLy*

**:FUNction<x>:POSition****Command/Query****DESCRIPTION**

This command sets the vertical position of the selected math operation (arithmetic and algebra operation).

This query returns the current position value for the selected operation.

**COMMAND SYNTAX**

:FUNction<x>:POSition <offset>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The range of values is uniform and related to an operation.

**QUERY SYNTAX**

:FUNction<x>:POSition?

**RESPONSE FORMAT**

<offset>

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command changes the vertical position of Function1 waveform to 1 V.

Command message:

*:FUNction1:POSition 5.00E-01*

*FUNC1:POS 5.00E-01*

Query message:

*FUNC1:POS?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:FUNction<x>:OPERation

**:FUNction<x>:SCALe****Command/Query****DESCRIPTION**

The command sets the vertical scale of the selected math operation (arithmetic and algebra operation).

The query returns the current scale value for the selected operation.

**COMMAND SYNTAX**

:FUNction<x>:SCALe <scale>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNction and defines the math that is affected by the command.

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

- The range of the function scale is related to the scale of the function source.
- When the operation is INTegrate and DIFF, the scale range is related to the timebase.

**QUERY SYNTAX**

:FUNction<x>:SCALe?

**RESPONSE FORMAT**

<scale>

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command changes the vertical scale of Function1 waveform to 1 V.

Command message:

*:FUNction1:SCALe 1.00E+00*  
*FUNC1:SCAL 1.00E+00*

Query message:

*FUNC1:SCAL?*

Response message:

*1.00E+00*

**RELATED COMMANDS**

:CHANnel<n>:SCALe

**:FUNCTION<x>:SOURCE1****Command/Query****DESCRIPTION**

This command sets the source1 of the math operation.

This query returns the current source1 of the math operation.

**COMMAND SYNTAX**

:FUNCTION<x>:SOURCE1 <source>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<source>:= {C<n>|Z<n>|F<x>|M<m>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function, for math-on-math operations.
- ◆ M denotes a memory waveform.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**Note:**

- Z<n> is optional only when Zoom is on.
- FUNCTION<x> cannot set itself as the source.

**QUERY SYNTAX**

:FUNCTION<x>:SOURCE1?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>|F<x>|M<m>}

**EXAMPLE**

The following command sets the source 1 of Function2 to C1.

Command message:

```
:FUNCTION2:SOURCE1 C1
FUNC2:SOUR1 C1
```

Query message:

```
FUNC2:SOUR1?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:FUNCTION<x>:SOURCE2

**:FUNCTION<x>:SOURCE2****Command/Query****DESCRIPTION**

This command sets the source2 of the math operation.

This query returns the current source2 of the math operation.

**COMMAND SYNTAX**

:FUNCTION<x>:SOURCE2 <source>

<x>:= 1 to (# math functions) in NR1 format, is attached as a suffix to FUNCTION and defines the math that is affected by the command.

<source>:= {C<n>|Z<n>|F<x>|M<m>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function, for math-on-math operations.
- ◆ M denotes a memory waveform.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**Note:**

- Z<n> is optional only when Zoom is on.
- FUNCTION<x> cannot set itself as the source.

**QUERY SYNTAX**

:FUNCTION<x>:SOURCE2?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>|F<x>|M<m>}

**EXAMPLE**

The following command sets the source2 of Function2 to C1.

Command message:

*:FUNCTION2:SOURCE2 C1*

*FUNC2:SOUR2 C1*

Query message:

*FUNC2:SOUR2?*

Response message:

*C1*

**RELATED COMMANDS**

:FUNCTION<x>:SOURCE1

## **HISTORY Commands**

The :HISTORY subsystem commands control the waveform recording function and the history waveform play function.

- ◆ **:HISTORY**
- ◆ **:HISTORY:FRAMe**
- ◆ **:HISTORY:INTERval**
- ◆ **:HISTORY:LIST**
- ◆ **:HISTORY:PLAY**
- ◆ **:HISTORY:TIME**



**:HISTORY****Command/Query****DESCRIPTION**

The command sets the mode of the history function.

This query returns the current status of the history function.

**COMMAND SYNTAX**

:HISTORY <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:HISTORY?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the history function.

Command message:

*:HISTORY ON*

*HISTOR ON*

Query message:

*HISTOR?*

Response message:

*ON*

**:HISTORy:FRAMe****Command/Query****DESCRIPTION**

This command sets the number of the history frame.

This query returns the current number of history frames.

**COMMAND SYNTAX**

:HISTORy:FRAMe <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

The maximum number of frames is related to the number of samples set for the acquisition (memory depth). More points/frame means less total frames available. Fewer points/frame equals more frames available.

**QUERY SYNTAX**

:HISTORy:FRAMe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of the history frame to 4.

Command message:

*:HISTORy:FRAMe 4*  
*HISTOR:FRAM 4*

Query message:

*HISTOR:FRAM?*

Response message:

*4*

**:HISTORy:INTERval****Command/Query****DESCRIPTION**

This command sets the play interval of the history frame.

This query returns the current play interval of the history frame.

**COMMAND SYNTAX**

:HISTORy:INTERval <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [1.00E-06, 1].

**QUERY SYNTAX**

:HISTORy:INTERval?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the play interval of the history frame to 1 ms.

Command message:

*:HISTORy:INTERval 1.00E-03*

*HISTOR:INTER 1.00E-03*

Query message:

*HISTOR:INTER?*

Response message:

*1.00E-03*

**:HISTORy:LIST****Command/Query****DESCRIPTION**

This command sets the state of the history list.

This query returns the current state of the history list.

**COMMAND SYNTAX**

:HISTORy:LIST <state>

<state>:= {OFF|ON[,<type>]}

<type>:= {TIME|DELTA}

- ◆ TIME indicates that the time column is displayed by sampling time
- ◆ DELTA indicates that the time column is displayed by the sampling interval.

**QUERY SYNTAX**

:HISTORy:LIST?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON[,<type>]}

<type>:= {TIME|DELTA}

**EXAMPLE**

The following command turns on the history list and displays it by sampling time.

Command message:

*:HISTORy:LIST ON,TIME*

*HISTOR:LIST ON,TIME*

Query message:

*HISTOR:LIST?*

Response message:

*ON,TIME*

**:HISTORy:PLAY****Command/Query****DESCRIPTION**

This command sets the play state of the history waveform.

This query returns the current play state of the history waveform.

**COMMAND SYNTAX**

:HISTORy:PLAY <state>

<state>:= {BACKWards|PAUSe|FORWards}

- ◆ BACKWards indicates that the frame number is played from highest frame number to lowest (last-to-first, chronologically).
- ◆ FORWards indicates that the frame number is played from the lowest frame number to the highest (first-to-last, chronologically).
- ◆ PAUSe will pause playback.

**QUERY SYNTAX**

:HISTORy:PLAY?

**RESPONSE FORMAT**

<state>

<state>:= {BACKWards|PAUSe|FORWards}

**EXAMPLE**

The following command sets the playback state of the history waveform to backwards.

Command message:

*:HISTORy:PLAY BACKWards*  
*HISTOR:PLAY BACKW*

Query message:

*HISTOR:PLAY?*

Response message:

*BACKWards*

**:HISTORy:TIME****Query****DESCRIPTION**

The query returns the acquire timestamp of the current frame.

**QUERY SYNTAX**

:HISTORy:TIME?

**RESPONSE FORMAT**

<time>

<time>:= hours:minutes:seconds.microseconds in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command returns the time of acquisition of the current frame.

Query message:

*:HISTOR:TIME?*

Response message:

*07:48:09.253827*

**RELATED COMMANDS**

:HISTORy:FRAMe

## MEASure Commands

The :MEASure subsystem commands are used to control automatic measurements.

- ◆ :MEASure
- ◆ :MEASure:ADVanced:CLEar
- ◆ :MEASure:ADVanced:LINenumber
- ◆ :MEASure:ADVanced:P<n>
- ◆ :MEASure:ADVanced:P<n>:SOURce1
- ◆ :MEASure:ADVanced:P<n>:SOURce2
- ◆ :MEASure:ADVanced:P<n>:STATistics
- ◆ :MEASure:ADVanced:P<n>:SHIStory
- ◆ :MEASure:ADVanced:P<n>:TYPE
- ◆ :MEASure:ADVanced:P<n>:VALue
- ◆ :MEASure:ADVanced:STATistics
- ◆ :MEASure:ADVanced:STATistics:AIMLimit
- ◆ :MEASure:ADVanced:STATistics:HISTOGram
- ◆ :MEASure:ADVanced:STATistics:MAXCount
- ◆ :MEASure:ADVanced:STATistics:RESet
- ◆ :MEASure:ADVanced:STYLE
- ◆ :MEASure:ASTRategy
- ◆ :MEASure:ASTRategy:BASE
- ◆ :MEASure:ASTRategy:TOP
- ◆ :MEASure:DITMe<n>:EDGE1
- ◆ :MEASure:DITMe<n>:EDGE2
- ◆ :MEASure:DITMe<n>:SLOPe1
- ◆ :MEASure:DITMe<n>:SLOPe2
- ◆ :MEASure:DITMe<n>:THReshold1
- ◆ :MEASure:DITMe<n>:THReshold2
- ◆ :MEASure:GATE
- ◆ :MEASure:GATE:GA
- ◆ :MEASure:GATE:GB
- ◆ :MEASure:MODE
- ◆ :MEASure:RDISplay
- ◆ :MEASure:SIMPlE:CLEar
- ◆ :MEASure:SIMPlE:ITEM
- ◆ :MEASure:SIMPlE:SOURce
- ◆ :MEASure:SIMPlE:VALue

- ◆ **:MEASure:THReshold:SOURce**
- ◆ **:MEASure:THReshold:TYPE**
- ◆ **:MEASure:THReshold:ABSolute**
- ◆ **:MEASure:THReshold:PERCent**



**:MEASure**

**Command/Query**

**DESCRIPTION**

The command sets the state of the measurement function.

This query returns the current state of the measurement function.

**COMMAND SYNTAX**

:MEASure <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEASure?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the measurement function.

Command message:

*:MEASure ON*

*:MEAS ON*

Query message:

*MEAS?*

Response message:

*ON*

**:MEASure:ADVanced:CLEar**

**Command**

**DESCRIPTION**

The command clears all the advanced measurement items.

**COMMAND SYNTAX**

:MEASure:ADVanced:CLEar

**EXAMPLE**

The following command clears the advanced measurement items.

Command message:

*:MEASure:ADVanced:CLEar*

*MEAS:ADV:CLE*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>

:MEASure:ADVanced:P<n>:TYPE

**:MEASure:ADVanced:LINenumber****Command/Query****DESCRIPTION**

The command sets the total number of advanced measurement items displayed.

The query returns the current total number of advanced measurement items displayed.

**COMMAND SYNTAX**

:MEASure:ADVanced:LINenumber <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 12].

**QUERY SYNTAX**

:MEASure:ADVanced:LINenumber?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the total number of advanced measurement items displayed to 12.

Command message:

```
:MEASure:ADVanced:LINenumber 12  
MEAS:ADV:LIN 12
```

Query message:

```
MEAS:ADV:LIN?
```

Response message:

```
12
```

**RELATED COMMANDS**

:MEASure:MODE

**:MEASure:ADVanced:P<n>****Command/Query****DESCRIPTION**

This command sets the state of the specified measurement item.

This query returns the current state of the measurement item.

**COMMAND SYNTAX**

:MEASure:ADVanced:P<n> <state>

P is the physical location of the specified measurement on the display.

<n>:= 1 to 12

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the first (leftmost/topmost) measurement item.

Command message:

*:MEASure:ADVanced:P1 ON*  
*MEAS:ADV:P1 ON*

Query message:

*MEAS:ADV:P1?*

Response message:

*ON*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>:TYPE  
:MEASure:ADVanced:P<n>:SOURce1  
:MEASure:ADVanced:P<n>:SOURce2

**:MEASure:ADVanced:P<n>:SOURce1****Command/Query****DESCRIPTION**

This command sets the source1 of the specified advanced measurement item.

This query returns the current source1 of the specified advanced measurement item.

**COMMAND SYNTAX**

:MEASure:ADVanced:P<n>:SOURce1 <source>

<n>:= 1 to 12

<source>:= {C<n>|Z<n>|F<x>|M<m>|D<d>|ZD<d>|REF<r>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ D denotes a digital channel.
- ◆ ZD denotes a zoomed digital channel.
- ◆ REF denotes a reference waveform.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

<r>:= {A|B|C|D}

**Note:**

- Z<n> and ZD<d> are optional only when Zoom is on.
- The source can only be set to C<n> or F<x> when the type is delay measurement.

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>:SOURce1?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>|F<x>|M<m>|D<d>|ZD<d>|REF<r>}

**EXAMPLE**

The following command sets the source1 of the first measurement item to C1.

Command message:

```
:MEASure:ADVanced:P1:SOURce1 C1
MEAS:ADV:P1:SOUR1 C1
```

Query message:

```
MEAS:ADV:P1:SOUR1?
```

Response message:

```
C1
```

**RELATED COMMANDS**

```
:MEASure:ADVanced:P<n>:SOURce2
:MEASure:ADVanced:P<n>:TYPE
```

**:MEASure:ADVanced:P<n>:SOURce2**

**Command/Query**

**DESCRIPTION**

This command sets the source2 of the specified advanced measurement item. When the measurement type is delay measurement item, source2 needs to be set.

This query returns the source2 of the specified advanced measurement item.

**COMMAND SYNTAX**

:MEASure:ADVanced:P<n>:SOURce2 <source>

<n>:= 1 to 12

<source>:= {C<n>|F<x>}

- ◆ C denotes an analog channel.
- ◆ F denotes a math function.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>:SOURce2?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|F<x>}

**EXAMPLE**

The following command sets the source2 of the first measurement item to C2.

Command message:

*:MEASure:ADVanced:P1:SOURce2 C2*  
*MEAS:ADV:P1:SOUR2 C2*

Query message:

*MEAS:ADV:P1:SOUR2?*

Response message:

*C2*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>:SOURce1  
 :MEASure:ADVanced:P<n>:TYPE

**:MEASure:ADVanced:P<n>:STATistics****Query****DESCRIPTION**

This query returns statistics for the specified advanced measurement item.

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>:STATistics? <type>

<n>:= 1 to 12

<type>:=

{ALL|CURRent|MEAN|MAXimum|MINimum|STDev|COUNT}

- ◆ ALL returns all the statistics
- ◆ CURRent returns the current value of the statistics
- ◆ MEAN returns the mean value of the statistics
- ◆ MAXimum returns the maximum value of the statistics
- ◆ MINimum returns the minimum value of the statistics
- ◆ STDev returns the standard deviation of the statistics
- ◆ COUNT returns the current number of counts used to calculate the statistical data

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

When measurement statistics are off, it returns OFF.

**EXAMPLE**

The following query returns the statistical current value of the first measurement item.

Query message:

*MEAS:ADV:P1:STAT? CURR*

Response message:

*6.7E-02*

**RELATED COMMANDS**

:MEASure:ADVanced:STATistics

**:MEASure:ADVanced:P<n>:SHIStory****Query****DESCRIPTION**

This query returns statistics for the specified advanced measurement item. Only valid when there is a limit on the maximum number of statistics, and returns the current statistical value when the count limit is Unlimited.

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>:SHIStory? [<count>]

<n>:= 1 to 12

<count>:= The number and range of historical statistical data are limited by the maximum number of statistics.

**Note:**

- When <count>is not specified, return all historical data.

**RESPONSE FORMAT**

Count =<value>,<value1>,<value2>,...,<valueN>

<value>:= Number of statistical data.

<valueN>:= Statistical data values.

**EXAMPLE**

When the maximum number of statistics is 1024, The following querys the top 5 measurement data of P1's current value.

Query message:

*MEAS:ADV:P1:SHIS? 5*

Response message:

*Count=5,4.223529E+00,4.221176E+00,4.221176E+00,  
4.214118E+00,4.223529E+00*

**RELATED COMMANDS**

:MEASure:ADVanced:STATistics:MAXCount

**:MEASure:ADVanced:P<n>:TYPE**

**Command/Query**

**DESCRIPTION**

This command sets the type for the specified measurement item.

This query returns the type for the specified measurement item.

**COMMAND SYNTAX**

:MEASure:ADVanced:P<n>:TYPE <parameter>

<n>:= 1 to 12

<parameter>:=

{PKPK|MAX|MIN|AMPL|TOP|BASE|LEVELX|CMEAN|MEAN|STDEV|VSTD|RMS|CRMS|MEDIAN|CMEDIAN|OVSN|FPRE|OVSP|RPRE|ULOWer|PER|FREQ|TMAX|TMIN|PWID|NWID|DUTY|NDUTY|WID|NBWID|DELAY|TIMEL|RISE|FALL|RISE10T90|FALL90T10|CCJ|PAREA|NAREA|AREA|ABSAREA|CYCLES|REDGES|FEDGES|EDGES|PPULSES|NPULSES|PHA|SKEW|FRR|FRF|FFR|FFF|LRR|LRF|LFR|LFF|PACArea|NACArea|ACArea|ABSACArea|PSLOPE|NSLOPE|TSR|TSF|THR|THF|DTIME1|DTIME2|DTIME3|DTIME4}

Description of Parameters

| Parameter | Description   |
|-----------|---|
| PKPK      | Difference between maximum and minimum data values  |
| MAX       | Highest value in waveform   |
| MIN       | Lowest value in waveform  |
| AMPL      | Difference between top and base in a bimodal waveform. If not bimodal, difference between max and min |
| TOP       | Value of most probable higher state in a bimodal waveform   |
| BASE      | Value of most probable lower state in a bimodal waveform  |
| LEVELX    | Level measured at trigger position  |
| CMEAN     | Average value of the first cycle  |
| MEAN      | Average of data values  |
| STDEV     | Standard deviation of the data  |
| VSTD      | Standard deviation of the first cycle   |
| RMS       | Root mean square of the data  |
| CRMS      | Root mean square of the first cycle   |
| MEDIAN    | Value at which 50% of the measurement are above and 50% are below                                     |
| CMEDIAN   | Median of the first cycle   |
| OVSN      | Overshoot following a falling edge; 100%*(base-min)/amplitude   |
| FPRE      | Overshoot before a falling edge; 100%*(max-top)/amplitude   |
| OVSP      | Overshoot following a rising edge; 100%*(max-top)/amplitude   |
| RPRE      | Overshoot before a rising edge; 100%*(base-min)/amplitude   |



|           |   |
|-----------|---|
| ULOWer    | The value between the threshold upper and lower   |
| PER       | Time between the middle threshold points of two consecutive, like-polarity edges  |
| FREQ      | Reciprocal of period  |
| TMAX      | First time of maximum value   |
| TMIN      | First time of minimum value   |
| PWID      | Time difference between the middle threshold of a rising edge to the middle threshold of the next falling edge of the pulse |
| NWID      | Time difference between the middle threshold of a falling edge to the middle threshold of the next rising edge of the pulse |
| DUTY      | Positive Duty Cycle. Ratio of positive width to period  |
| NDUTY     | Duty Cycle. Ratio of negative width to period   |
| WID       | Time from the first rising edge to the last falling edge at the middle threshold  |
| NBWID     | Time from the first falling edge to the last rising edge at the middle threshold  |
| DELAY     | Time from the trigger to the first transition at the middle threshold   |
| TIMEL     | Time from the trigger to each rising edge at the middle threshold   |
| RISE      | Duration of rising edge from lower to upper of threshold  |
| FALL      | Duration of falling edge from upper to lower of threshold   |
| RISE10T90 | Duration of rising edge from 10-90%   |
| FALL90T10 | Duration of falling edge from 90-10%  |
| CCJ       | The difference between two continuous periods   |
| PAREA     | Area of the waveform above zero   |
| NAREA     | Area of the waveform below zero   |
| AREA      | Area of the waveform  |
| ABSAREA   | Absolute area of the waveform   |
| CYCLES    | Number of cycles in a periodic waveform   |
| EDGES     | Number of edges in a waveform   |
| REDGES    | Number of rising edges in a waveform  |
| FEDGES    | Number of falling edges in a waveform   |
| PPULSES   | Number of positive pulses in a waveform   |
| NPULSES   | Number of negative pulses in a waveform   |
| PHA       | Phase difference between two edges  |
| SKEW      | Time of source A edge minus time of nearest source B edge   |
| FRR       | The time between the first rising edge of source A and the first rising edge of source B at the middle threshold            |
| FRF       | The time between the first rising edge of source A and the first falling edge of source B at the middle threshold           |
| FFR       | The time between the first falling edge of  |

|           |  |
|-----------|--|
|           | source A and the first rising edge of source B at the middle threshold   |
| FFF       | The time between the first falling edge of source A and the first falling edge of source B at the middle threshold |
| LRR       | The time between the first rising edge of source A and the last rising edge of source B at the middle threshold    |
| LRF       | The time between the first rising edge of source A and the last falling edge of source B at the middle threshold   |
| LFR       | The time between the first falling edge of source A and the last rising edge of source B at the middle threshold   |
| LFF       | The time between the first falling edge of source A and the last falling edge of source B at the middle threshold  |
| PACArea   | Area of the waveform above average   |
| NACArea   | Area of the waveform below average   |
| ACArea    | Area of the waveform above average minus area of the waveform below average  |
| ABSACArea | Area of the waveform above average add area of the waveform below average  |
| PSLOPE    | The slope of rising edges  |
| NSLOPE    | The slope of falling edges   |
| TSR       | Data setup time before the clock rising edge   |
| TSF       | Data setup time before the clock falling edge  |
| THR       | Data hold time after the clock rising edge   |
| THF       | Data hold time after the clock falling edge  |
| DITMe1-4  | The time difference between the specified two edges  |

**QUERY SYNTAX**

:MEASure:ADVanced:P&lt;n&gt;:TYPE?

**RESPONSE FORMAT**

&lt;parameter&gt;

**EXAMPLE**

The following command sets the type of the first measurement to maximum.

Command message:

```
:MEASure:ADVanced:P1:TYPE MAX
MEAS:ADV:P1:TYPE MAX
```

Query message:

```
MEAS:ADV:P1:TYPE?
```

Response message:

```
MAX
```

**RELATED COMMANDS**

:MEASure:ADVanced:P&lt;n&gt;

**:MEASure:ADVanced:P<n>:VALue****Query****DESCRIPTION**

The query returns the value of the specified advanced measurement item.

**QUERY SYNTAX**

:MEASure:ADVanced:P<n>:VALue?

<n>:= 1 to 12

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following query returns the value of the first measurement item.

Query message:

*MEAS:ADV:P1:VAL?*

Response message:

*4.033E+00*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>:TYPE

**:MEASure:ADVanced:STATistics****Command/Query****DESCRIPTION**

The command sets the state of the measurement statistics.

This query returns the current state of the measurement statistics function.

**COMMAND SYNTAX**

:MEASure:ADVanced:STATistics <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEASure:ADVanced:STATistics?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the statistics function.

Command message:

*:MEASure:ADVanced:STATistics ON*

*MEAS:ADV:STAT ON*

Query message:

*MEAS:ADV:STAT?*

Response message:

*ON*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>:STATistics

**:MEASure:ADVanced:STATistics:AIMLimit****Command/Query****DESCRIPTION**

The command sets the value of the measurement statistics AIM limit.

This query returns the current value of the measurement statistics AIM limit.

**COMMAND SYNTAX**

:MEASure:ADVanced:STATistics:AIMLimit <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:MEASure:ADVanced:STATistics:AIMLimit?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the statistics aim limit to 500.

Command message:

*:MEASure:ADVanced:STATistics: AIMLimit 500*  
*MEAS:ADV:STAT:AIML 500*

Query message:

*MEAS:ADV:STAT:AIML?*

Response message:

*500*

**RELATED COMMANDS**

:MEASure:ADVanced:P<n>:STATistics

**:MEASure:ADVanced:STATistics:HISTOGRAM****Command/Query****DESCRIPTION**

The command sets the state of the histogram function.

This query returns the current state of the histogram function.

**COMMAND SYNTAX**

:MEASure:ADVanced:STATistics:HISTOGRAM <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEASure:ADVanced:STATistics:HISTOGRAM?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables histogram function.

Command message:

*:MEASure:ADVanced:STATistics:HISTOGRAM ON*  
*MEAS:ADV:STAT:HISTOG ON*

Query message:

*MEAS:ADV:STAT:HISTOG?*

Response message:

*ON*

**RELATED COMMANDS**

:MEASure:ADVanced:STATistics

**:MEASure:ADVanced:STATistics:MAXCount****Command/Query****DESCRIPTION**

This command sets the maximum value of the statistics count.

The query returns the current value of statistics count.

**COMMAND SYNTAX**

:MEASure:ADVanced:STATistics:MAXCount <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 1024].

Note:

When the value is set to 0, it means unlimited statistics.

**QUERY SYNTAX**

:MEASure:ADVanced:STATistics:MAXCount?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the maximum value of statistics count to 1024.

Command message:

*:MEASure:ADVanced:STATistics:MAXCount 1024*

*MEAS:ADV:STAT:MAXC 1024*

Query message:

*MEAS:ADV:STAT:MAXC?*

Response message:

*1024*

**RELATED COMMANDS**

:MEASure:ADVanced:STATistics

**:MEASure:ADVanced:STATistics:RESet****Command**

**DESCRIPTION** The command resets the measurement statistics.

**COMMAND SYNTAX** :MEASure:ADVanced:STATistics:RESet

**EXAMPLE** The following command restarts statistics.

Command message:

```
:MEASure:ADVanced:STATistics:RESet  
MEAS:ADV:STAT:RES
```

**RELATED COMMANDS** :MEASure:ADVanced:STATistics

**:MEASure:ADVanced:STYLE****Command/Query**

**DESCRIPTION** The command selects the display mode of the advanced measurements.

This query returns the current display mode of the advanced measurement.

**COMMAND SYNTAX** :MEASure:ADVanced:STYLE <type>

<type>:= {M1|M2}

- ◆ M1 lists a measurement, corresponding statistics, and histogram vertically on the display.
- ◆ M2 lists a measurement and corresponding statistics horizontally on the display. No histogram is available with M2.

**QUERY SYNTAX** :MEASure:ADVanced:STYLE?

**RESPONSE FORMAT** <type>

<type>:= {M1|M2}

**EXAMPLE** The following command selects the display mode of the advanced measurement to M1.

Command message:

```
:MEASure:ADVanced:STYLE M1  
MEAS:ADV:STYL M1
```

Query message:

```
MEAS:ADV:STYL?
```

Response message:

```
M1
```



**:MEASure:ASTRategy****Command/Query****DESCRIPTION**

The command sets the mode of amplitude calculation strategy.

This query returns the current mode of the amplitude calculation strategy.

**COMMAND SYNTAX**

:MEASure:ASTRategy <type>

<type>:= {AUTO|MANual}

- ◆ AUTO sets the amplitude calculation strategy will be selected automatically according to the input signal to ensure the accuracy of the measured value.
- ◆ MANual sets the amplitude calculation strategy will be selected manually.

**QUERY SYNTAX**

:MEASure:ASTRategy?

**RESPONSE FORMAT**

<type>

<type>:= {AUTO|MANual}

**EXAMPLE**

The following command selects the amplitude calculation strategy to MANual.

Command message:

*:MEASure:ASTRategy MANual*  
*MEAS:ASTR MAN*

Query message:

*MEAS:ASTR?*

Response message:

*MANual*

**:MEASure:ASTRategy:BASE****Command/Query****DESCRIPTION**

The command sets the mode of amplitude calculation base strategy.

This query returns the current mode of the amplitude calculation base strategy.

**COMMAND SYNTAX**

:MEASure:ASTRategy:BASE <type>

<type>:= {HISTogram|MIN}

- ◆ HISTogram sets the amplitude calculation base strategy will identify the value with the maximum probability as the base value.
- ◆ MIN sets the amplitude calculation top strategy will identify the minimum value of the waveform as the base value.

**QUERY SYNTAX**

:MEASure:ASTRategy:BASE?

**RESPONSE FORMAT**

<type>

<type>:= {HISTogram|MIN}

**EXAMPLE**

The following command selects the amplitude calculation base strategy to Histogram.

Command message:

*:MEASure:ASTRategy:BASE HISTogram*  
*MEAS:ASTR:BASE HIST*

Query message:

*MEAS:ASTR:BASE?*

Response message:

*HISTogram*

**:MEASure:ASTRategy:TOP****Command/Query****DESCRIPTION**

The command sets the mode of amplitude calculation top strategy.

This query returns the current mode of the amplitude calculation top strategy.

**COMMAND SYNTAX**

:MEASure:ASTRategy:TOP <type>

<type>:= {HISTogram|MAX}

- ◆ HISTogram sets the amplitude calculation top strategy will identify the value with the maximum probability as the top value.
- ◆ MAX sets the amplitude calculation top strategy will identify the maximum value of the waveform as the top value.

**QUERY SYNTAX**

:MEASure:ASTRategy:TOP?

**RESPONSE FORMAT**

<type>

<type>:= {HISTogram|MAX}

**EXAMPLE**

The following command selects the amplitude calculation top strategy to Histogram.

Command message:

*:MEASure:ASTRategy:TOP HISTogram*  
*MEAS:ASTR:TOP HIST*

Query message:

*MEAS:ASTR:TOP?*

Response message:

*HISTogram*

**:MEASure:DITMe<n>:EDGE1****Command/Query****DESCRIPTION**

This command sets the starting edge of the specified delta time.

This query returns the starting edge of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:EDGE1 <value>

<n>:= delta time index, the range is [1,4]

<value>:= start edge in NR1 format. The value range is limited by the AIM limit. When set to -1, it indicates the last edge

**QUERY SYNTAX**

:MEASure:DITMe<n>:EDGE1?

**RESPONSE FORMAT**

<value>

**EXAMPLE**

The following command sets the starting edge of delta time1 as the last edge.

Command message:

```
:MEASure:DITMe1:EDGE1 -1  
MEAS:DITM1:EDGE1 -1
```

Query message:

```
MEAS:DITM1:EDGE1?
```

Response message:

```
-1
```

**RELATED COMMANDS**

```
:MEASure:ADVanced:STATistics:AIMLimit  
:MEASure:DITMe<n>:EDGE2
```

**:MEASure:DITMe<n>:EDGE2**

**Command/Query**

**DESCRIPTION**

This command sets the end edge of the specified delta time.

This query returns the end edge of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:EDGE2 <value>

<n>:= delta time index, the range is [1,4].

<value>:= end edge in NR1 format. The value range is limited by the AIM limit. When set to -1, it indicates the last edge

**QUERY SYNTAX**

:MEASure:DITMe<n>:EDGE2?

**RESPONSE FORMAT**

<value>

**EXAMPLE**

The following command sets the end edge of delta time 1 as the last edge.

Command message:

*:MEASure:DITMe1:EDGE2 -1*

*MEAS:DITM1:EDGE2 -1*

Query message:

*MEAS:DITM1:EDGE2?*

Response message:

*-1*

**RELATED COMMANDS**

:MEASure:ADVanced:STATistics:AIMLimit

:MEASure:DITMe<n>:EDGE1

**:MEASure:DITMe<n>:SLOPe1****Command/Query****DESCRIPTION**

This command sets the start edge slope type of the specified delta time.

This query returns the start edge slope type of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:SLOPe1 <slope\_type>

<n>:= delta time index, the range is [1,4]

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:MEASure:DITMe<n>:SLOPe1?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the start edge slope of delta time1 to RISing.

Command message:

*:MEASure:DITMe1:SLOPe1 RISing*  
*MEAS:DITM1:SLOP1 RIS*

Query message:

*MEAS:DITM1:SLOP1?*

Response message:

*RISing*

**RELATED COMMANDS**

:MEASure:DITMe<n>:SLOPe2

**:MEASure:DITMe<n>:SLOPe2****Command/Query****DESCRIPTION**

This command sets the end edge slope type of the specified delta time.

This query returns the end edge slope type of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:SLOPe2 <slope\_type>

<n>:= delta time index, the range is [1,4].

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:MEASure:DITMe<n>:SLOPe2?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the end edge slope of delta time1 to RISing.

Command message:

*:MEASure:DITMe1:SLOPe2 RISing*  
*MEAS:DITM1:SLOP2 RIS*

Query message:

*MEAS:DITM1:SLOP2?*

Response message:

*RIS*

**RELATED COMMANDS**

:MEASure:DITMe<n>:SLOPe1

**:MEASure:DITMe<n>:THReshold1****Command/Query****DESCRIPTION**

This command sets the threshold type for the start edge of the specified delta time.

This query returns the threshold type for the start edge of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:THReshold1 <type>

<n>:= delta time index, the range is [1,4].

<type>:= {UPPer|MIDDLE|LOWer}

**QUERY SYNTAX**

:MEASure:DITMe<n>:THReshold1?

**RESPONSE FORMAT**

<type>

<type>:= {UPPer|MIDDLE|LOWer}

**EXAMPLE**

The following command sets the threshold type of the start edge of delta time1 to UPPer.

Command message:

*:MEASure:DITMe1:THReshold1 UPPer*  
*MEAS:DITM1:THR1 UPP*

Query message:

*MEAS:DITM1:THR1?*

Response message:

*UPPer*

**RELATED COMMANDS**

:MEASure:DITMe<n>:THReshold2



**:MEASure:DITMe<n>:THReshold2**

**Command/Query**

**DESCRIPTION**

This command sets the threshold type for the end edge of the specified delta time.

This query returns the threshold type for the end edge of the specified delta time.

**COMMAND SYNTAX**

:MEASure:DITMe<n>:THReshold2 <type>

<n>:= delta time index, the range is [1,4].

<type>:= {UPPer|MIDDLE|LOWer}

**QUERY SYNTAX**

:MEASure:DITMe<n>:THReshold2?

**RESPONSE FORMAT**

<type>

<type>:= {UPPer|MIDDLE|LOWer}

**EXAMPLE**

The following command sets the threshold type of the end edge of delta time1 to UPPer.

Command message:

*:MEASure:DITMe1:THReshold2 UPPer  
MEAS:DITM1:THR2 UPP*

Query message:

*MEAS:DITM1:THR2?*

Response message:

*UPPer*

**RELATED COMMANDS**

:MEASure:DITMe<n>:THReshold1

**:MEASure:GATE****Command/Query****DESCRIPTION**

This command sets the state of the measurement gate.

This query returns the current state of the measurement gate.

**COMMAND SYNTAX**

:MEASure:GATE <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEASure:GATE?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the measurement gate.

Command message:

*:MEASure:GATE ON*

*MEAS:GATE ON*

Query message:

*MEAS:GATE?*

Response message:

*ON*

**RELATED COMMANDS**

:MEASure:GATE:GA

:MEASure:GATE:GB

**:MEASure:GATE:GA****Command/Query****DESCRIPTION**

This command sets the position of gate A.

This query returns the current position of gate A.

**COMMAND SYNTAX**

:MEASure:GATE:GA <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-horizontal\_grid/2\*timebase, horizontal\_grid/2\*timebase].

**Note:**

The value of GA cannot be greater than that of GB. If you set the value greater than GB, it will automatically be set to the same value as GB.

**QUERY SYNTAX**

:MEASure:GATE:GA?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the position of gate A to -100 ns.

Command message:

*:MEASure:GATE:GA -1.00E-07*

*MEAS:GATE:GA -1.00E-07*

Query message:

*MEAS:GATE:GA?*

Response message:

*-1.00E-07*

**RELATED COMMANDS**

:MEASure:GATE

:MEASure:GATE:GB

**:MEASure:GATE:GB****Command/Query****DESCRIPTION**

This command sets the position of gate B.

This command returns the current position of gate B.

**COMMAND SYNTAX**

:MEASure:GATE:GB <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-horizontal\_grid/2\*timebase, horizontal\_grid/2\*timebase].

**Note:**

The value of GB cannot be less than that of GA. If you set the value less than GA, it will automatically be set to the same value as GA.

**QUERY SYNTAX**

:MEASure:GATE:GB?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the position of gate B to 100 ns.

Command message:

*:MEASure:GATE:GB 1.00E-07*

*MEAS:GATE:GB 1.00E-07*

Query message:

*MEAS:GATE:GB?*

Response message:

*1.00E-07*

**RELATED COMMANDS**

:MEASure:GATE

:MEASure:GATE:GA

**:MEASure:MODE****Command/Query****DESCRIPTION**

The command specifies the mode of measurement.

The query returns the current mode of measurement.

**COMMAND SYNTAX**

:MEASure:MODE <type>

<type>:= {SIMPlE|ADVanced}

- ◆ SIMPlE shows measurements only
- ◆ ADVanced shows measurements and includes selections for statistics, view mode (M1, M2), histogram, and trending.

**QUERY SYNTAX**

:MEASure:MODE?

**RESPONSE FORMAT**

<type>

<type>:= {SIMPlE|ADVanced}

**EXAMPLE**

The following command sets the measurement mode to simple.

Command message:

*:MEASure:MODE SIMPlE*

*MEAS:MODE SIMP*

Query message:

*MEAS:MODE?*

Response message:

*SIMPlE*

**:MEASure:RDISplay****Command/Query****DESCRIPTION**

The command sets the display style of measurement results.

The query returns the display style of measurement results.

**COMMAND SYNTAX**

:MEASure:RDISplay <type>

<type>:= {EMBedded|FLOating}

- ◆ EMBedded means that the measurement results are embedded in the screen, which compresses the waveform area.
- ◆ FLOating means that the measurement results are suspended on the screen and will not compress the waveform area.

**QUERY SYNTAX**

:MEASure:RDISplay?

**RESPONSE FORMAT**

<type>

<type>:= {EMBedded|FLOating}

**EXAMPLE**

The following command sets the measurement result display style to FLOating.

Command message:

*:MEASure:RDISplay FLOating*  
*MEAS:RDIS FLO*

Query message:

*MEAS:RDIS?*

Response message:

*FLOating*

**:MEASure:SIMPlE:CLEar**

**Command**

**DESCRIPTION** The command clears all the simple measurement item.

**COMMAND SYNTAX** :MEASure:SIMPlE:CLEar

**EXAMPLE** The following command clears the simple measurement item.

Command message:  
*:MEASure:SIMPlE:CLEar*  
*MEAS:SIMP:CLE*

**RELATED COMMANDS** :MEASure:SIMPlE:ITEM

**:MEASure:SIMPlE:ITEM**

**Command**

**DESCRIPTION** This command sets the type of simple measurement.

**COMMAND SYNTAX** :MEASure:SIMPlE:ITEM <parameter>,<state>

<parameter>:=  
 {PKPK|MAX|MIN|AMPL|TOP|BASE|LEVELX|CMEAN|MEAN|STDEV|VSTD|RMS|CRMS|MEDIAN|CMEDIAN|OVSN|ULOWER|FPRE|OVSP|RPRE|PER|FREQ|TMAX|TMIN|PWID|NWID|DUTY|NDUTY|WID|NBWID|DELAY|TIMEL|RISE|FALL|RISE20T80|FALL80T20|CCJ|PAREA|NAREA|AREA|ABSAREA|CYCLES|REDGES|FEDGES|EDGES|PPULSES|NPULSES|PACArea|NACArea|ACArea|ABSACArea}

<state>:= {ON|OFF}

**Note:**  
 See the table Description of Parameters  
 Description of Parameters  
 Description of Parameters  
 Description of Parameters  
 Description of Parameters

**EXAMPLE** The following command adds maximum to the simple measurements window.

Command message:  
*:MEASure:SIMPlE:ITEM MAX,ON*  
*MEAS:SIMP:ITEM MAX,ON*

**RELATED COMMANDS** :MEASure:SIMPlE:VALue

**:MEASure:SIMPlE:SOURce****Command/Query****DESCRIPTION**

This command sets the source of the simple measurement.

This query returns the current source of the simple measurement.

**COMMAND SYNTAX**

:MEASure:SIMPlE:SOURce <source>

<source>:= {C<n>|Z<n>|F<x >|ZF<x>|M<m>|ZM<m>|D<d>|ZD<d>|REF<r>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ D denotes a digital channel.
- ◆ REF denotes a reference waveform.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

<r>:= {A|B|C|D}

**Note:**

Z<n>, ZF<x>, ZM<m> and ZD<d> are optional only when Zoom is on.

**QUERY SYNTAX**

:MEASure:SIMPlE:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>|F<x >|ZF<x>|M<m>|ZM<m>|D<d>|ZD<d>|REF<r>}

**EXAMPLE**

The following command sets the source of simple measurement to C1.

Command message:

```
:MEASure:SIMPlE:SOURce C1
MEAS:SIMP:SOUR C1
```

Query message:

```
MEAS:SIMP:SOUR?
```

Response message:

```
C1
```



**:MEASure:SIMPlE:VALue**

**Query**

**DESCRIPTION**

This query returns the specified measurement value that appears on the simple measurement.

**QUERY SYNTAX**

:MEASure:SIMPlE:VALue? <type>

<type>:=  
 {PKPK|MAX|MIN|AMPL|TOP|BASE|LEVELX|CMEAN|MEAN|  
 STDEV|VSTD|RMS|CRMS|MEDIAN|CMEDIAN|OVSN|FPRE|  
 OVSP|RPRE|ULOWer|PER|FREQ|TMAX|TMIN|PWID|NWID|D  
 UTY|NDUTY|WID|NBWID|DELAY|TIMEL|RISE|FALL|RISE20T  
 80|FALL80T20|CCJ|PAREA|NAREA|AREA|ABSAREA|CYCLE  
 S|REDGES|FEDGES|EDGES|PPULSES|NPULSES|PACArea|  
 NACArea|ACArea|ABSACArea|ALL}

**Note:**

See the table Description of Parameters

Description of Parameters

Description of Parameters

- ALL is only valid for queries, and it returns all measurement values of all measurement types except for delay measurements.

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following query returns the maximum value.

Query message:

*MEAS:SIMP:VAL? MAX*

Response message:

*2.000E+00*

**RELATED COMMANDS**

:MEASure:SIMPlE:ITEM

**:MEASure:THReshold:SOURce****Command/Query****DESCRIPTION**

This command sets the measurement threshold source.

This query returns the current measurement threshold source.

**COMMAND SYNTAX**

:MEASure:THReshold:SOURce <source>

<source>:= {C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|REF<r>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ REF denotes a reference waveform.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<r>:= {A|B|C|D}

**Note:**

Z<n>, ZF<x> and ZM<m> is optional only when Zoom is on.

**QUERY SYNTAX**

:MEASure:THReshold:SOURce?

**RESPONSE FORMAT**

<source>:= {C<n>|Z<n>|F<x>|ZF<x>|M<m>|ZM<m>|REF<r>}

**EXAMPLE**

The following command sets the threshold source to C1.

Command message:

```
:MEASure:THReshold:SOURce C1
MEAS:THR:SOUR C1
```

Query message:

```
MEAS:THR:SOUR?
```

Response message:

```
C1
```

**:MEASure:THReshold:TYPE**

**Command/Query**

**DESCRIPTION**

This command sets the measurement threshold type.

This query returns the current measurement threshold type.

**COMMAND SYNTAX**

:MEASure:THReshold:TYPE <type>

<type>:= {PERCent|ABSolute}

**QUERY SYNTAX**

:MEASure:THReshold:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {PERCent|ABSolute}

**EXAMPLE**

The following command sets the threshold typr to percent.

Command message:

*:MEASure:THReshold:TYPE PERCent*  
*MEAS:THR:TYPE PERC*

Query message:

*MEAS:THR:TYPE?*

Response message:

*PERCent*

**:MEASure:THReshold:ABSolute****Command/Query****DESCRIPTION**

This command specifies the reference level when :MEASure:THReshold:TYPE is set to ABSolute. This command affects the results of some measurements.

This query returns the reference level of the source

**COMMAND SYNTAX**

:MEASure:THReshold:ABSolute <high>,<mid>,<low>

<high>,<mid>,<low>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:MEASure:THReshold:ABSolute?

**RESPONSE FORMAT**

<high>,<mid>,<low>

<high>,<mid>,<low>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper,middle and lower threshold to 3V,1V,-1.5V.

Command message:

```
:MEASure:THReshold:ABSolute 3.00,1.00,-1.50  
MEAS:THR:ABS 3.00,1.00,-1.50
```

Query message:

```
MEAS:THR:ABS?
```

Response message:

```
3.00 E+00,1.00 E+00,-1.50E+00
```

**RELATED COMMANDS**

```
:MEASure:THReshold:TYPE  
:MEASure:SIMPlE:ITEM
```

**:MEASure:THReshold:PERCent****Command/Query****DESCRIPTION**

This command specifies the percent used to calculate the reference level when :MEASure:THReshold:TYPE is set to PERCent. This command affects the results of some measurements.

**COMMAND SYNTAX**

:MEASure:THReshold:PERCent <high>,<mid>,<low>

<high>,<mid>,<low>:= Value in NR1 format, including an integer and no decimal point, like 10

**QUERY SYNTAX**

:MEASure:THReshold:PERCent?

**RESPONSE FORMAT**

<high>,<mid>,<low>

<high>,<mid>,<low>:= Value in NR1 format, including an integer and no decimal point, like 10

**EXAMPLE**

The following command sets the upper,middle and lower threshold to 80%,45%,10%.

Command message:

*:MEASure:THReshold:PERCent 80,45,10*

*MEAS:THR:PERC 80,45,10*

Query message:

*MEAS:THR:PERC?*

Response message:

*80,45,10*

**RELATED COMMANDS**

:MEASure:SIMPle:ITEM

## MEMory Commands

The MEMory subsystem commands control memory waveforms.

- ◆ :MEMory<m>:HORizontal:POSition
- ◆ :MEMory<m>:HORizontal:SCALE
- ◆ :MEMory<m>:HORizontal:SYNC
- ◆ :MEMory<m>:IMPort
- ◆ :MEMory<m>:LABel
- ◆ :MEMory<m>:LABel:TEXT
- ◆ :MEMory<m>:SWITCh
- ◆ :MEMory<m>:VERTical:POSition
- ◆ :MEMory<m>:VERTical:SCALE

**:MEMory<m>:HORizontal:POSition****Command/Query****DESCRIPTION**

The command specifies the horizontal position of the memory waveform.

The query returns the current horizontal position of the memory.

**COMMAND SYNTAX**

:MEMory<m>:HORizontal:POSition <val>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<val>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:MEMory<m>:HORizontal:POSition?

**RESPONSE FORMAT**

<val>

<val>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command specifies a 10 us delay of M2 to the trigger point.

Command message:

*:MEMory2:HORizontal:POSition 1.00E-05*  
*MEM2:HOR:POS 1.00E-05*

Query message:

*MEM2:HOR:POS?*

Response message:

*1.00E-05*

**:MEMory<m>:HORizontal:SCALe****Command/Query****DESCRIPTION**

The command sets the horizontal scale per division for the memory waveform.

The query returns the current horizontal scale setting in seconds per division for the memory.

**COMMAND SYNTAX**

:MEMory<m>:HORizontal:SCALe <value>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:MEMory<m>:HORizontal:SCALe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command horizontal scale of M2 to 100 ns/div.

Command message:

```
:MEMory2:HORizontal:SCALe 1.00E-07  
MEM2:HOR:SCAL 1.00E-07
```

Query message:

```
MEM2:HOR:SCAL?
```

Response message:

```
1.00E-07
```



**:MEMory<m>:HORizontal:SYNC****Command/Query****DESCRIPTION**

The command turns on and off the horizontal parameter synchronization switch. When enabled, modify the horizontal parameters of the imported source, and the parameters of its memory waveform will also be modified synchronously.

This query returns the current state of the horizontal parameter synchronization switch.

**COMMAND SYNTAX**

:MEMory<m>:HORizontal:SYNC <state>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEMory<m>:HORizontal:SYNC?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the horizontal parameter synchronization switch of M2.

Command message:

*:MEMory2:HORizontal:SYNC ON*  
*MEM2:HOR:SYNC ON*

Query message:

*MEM2:HOR:SYNC?*

Response message:

*ON*

**:MEMory<m>:IMPort****Command****DESCRIPTION**

The command import the source to the memory waveform.

**COMMAND SYNTAX**

:MEMory<m>:IMPort <source>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<source>:= {C<n>|Z<n>|F<x>|M<m>|<path>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ <path>:= Quoted string of path with an extension “.bin”, denotes a waveform file.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command imports waveform of C2 to the M2.

Command message:

```
:MEMory2:IMPort C2
MEM2:IMP C2
```

**:MEMory<m>:LABel****Command/Query****DESCRIPTION**

The command is to turn the specified memory label on or off.

This query returns the label associated with a particular memory function.

**COMMAND SYNTAX**

:MEMory<m>:LABel <state>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MEMory<m>:LABel?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the label of M2.

Command message:

*:MEMory2:LABel ON*

*MEM2:LAB ON*

Query message:

*MEM2:LAB?*

Response message:

*ON*

**:MEMory<m>:LABel:TEXT****Command/Query****DESCRIPTION**

The command sets the selected memory label to the string that follows. Setting a label for a memory waveform also adds the name to the label list in non-volatile memory (replacing the oldest label in the list)

The query returns the current label text of the selected memory waveform.

**COMMAND SYNTAX**

:MEMory<m>:LABel:TEXT <string>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<string>:= Quoted string of ASCII text. The length of the string is limited to 20.

**QUERY SYNTAX**

:MEMory<m>:LABel:TEXT?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the label text of the M2 to "MATH".

Command message:

```
:MEMory2:LABel:TEXT "MATH"
MEM2:LAB:TEXT "MATH"
```

Query message:

```
MEM2:LAB:TEXT?
```

Response message:

```
"MATH"
```

## **:MEMory<m>:SWITCh**

### **Command**

#### **DESCRIPTION**

The command sets the display of the memory waveform.

This query returns the current display of the memory waveform.

#### **COMMAND SYNTAX**

**:MEMory<m>:SWITCh <state>**

**<m>:=** 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

**<state>:=** {ON|OFF}

#### **QUERY SYNTAX**

**:MEMory<m>:SWITCh?**

#### **RESPONSE FORMAT**

**<state>**

**<state>:=** {ON|OFF}

#### **EXAMPLE**

The following command enables the display of the M2.

Command message:

```
:MEMory2:SWITCh ON
MEM2:SWIT ON
```

Query message:

```
MEM2:SWIT?
```

Response message:

```
ON
```

**:MEMory<m>:VERTical:POSition****Command/Query****DESCRIPTION**

The command the vertical position of the selected memory waveform.

This query returns the current position value for the selected memory.

**COMMAND SYNTAX**

:MEMory<m>:VERTical:POSition <offset>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:MEMory<m>:VERTical:POSition?

**RESPONSE FORMAT**

<offset>

<offset>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command changes the vertical position of M2 waveform to 100m V.

Command message:

```
:MEMory2:VERTical:POSition 1.00E-01
MEM2:VERT:POS 1.00E-01
```

Query message:

```
MEM2:VERT:POS?
```

Response message:

```
1.00E-01
```

**:MEMory<m>:VERTical:SCALe****Command/Query****DESCRIPTION**

The command sets the vertical scale of the selected memory waveform.

The query returns the current scale value for the selected memory waveform.

**COMMAND SYNTAX**

:MEMory<m>:VERTical:SCALe <scale>

<m>:= 1 to (# memory waveforms) in NR1 format, including an integer and no decimal point, like 1.

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**QUERY SYNTAX**

:MEMory<m>:VERTical:SCALe?

**RESPONSE FORMAT**

<scale>

<scale>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command changes the vertical scale of M2 waveform to 100m V.

Command message:

*:MEMory2:VERTical:SCALe 1.00E-01*  
*MEM2:VERT:SCAL 1.00E-01*

Query message:

*MEM2:VERT:SCAL?*

Response message:

*1.00E-01*

## MTEst Commands

The :MTEst subsystem commands control the mask test features.

- ◆ :MTESt
- ◆ :MTESt:COUNt
- ◆ :MTESt:FUNcTion:BUZZer
- ◆ :MTESt:FUNcTion:COF
- ◆ :MTESt:FUNcTion:FTH
- ◆ :MTESt:FUNcTion:SOF
- ◆ :MTESt:IDISplay
- ◆ :MTESt:MASK:CREate
- ◆ :MTESt:MASK:LOAD
- ◆ :MTESt:OPERate
- ◆ :MTESt:RESet
- ◆ :MTESt:SOURce
- ◆ :MTESt:TYPE



**:MTESt****Command/Query****DESCRIPTION**

The command sets the state of the mask test.

This query returns the current state of the mask test.

**COMMAND SYNTAX**

:MTESt <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the mask test function.

Command message:

*:MTESt ON*  
*MTESt ON*

Query message:

*MTESt?*

Response message:

*ON*

**:MTESt:COUNT****Query****DESCRIPTION**

The query returns the result of the mask test.

**QUERY SYNTAX**

:MTESt:COUNT?

**RESPONSE FORMAT**

FAIL,<num>,PASS,<num>,TOTAL,<num>

<num>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command returns the count of the mask test.

Query message:

*MTESt:COUNT?*

Response message:

*FAIL,38176,PASS,5617,TOTAL,43793*

**RELATED COMMANDS**

:MTESt:OPERate

**:MTESt:FUNCtion:BUZZer****Command/Query****DESCRIPTION**

This command sets the state of the buzzer when failure frames are detected.

This command query returns the status of the buzzer.

**COMMAND SYNTAX**

:MTESt:FUNCtion:BUZZer <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt:FUNCtion:BUZZer?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the buzzer.

Command message:

*:MTESt:FUNCtion:BUZZer ON*

*MTESt:FUNC:BUZZ ON*

Query message:

*MTESt:FUNC:BUZZ?*

Response message:

*ON*

**:MTESt:FUNCtion:COF****Command/Query****DESCRIPTION**

This command sets the state of the mask test function "Capture on Fail". When this function is enabled, the default path to save the image of failing frames is "SIGLENT/".

This command query returns the status of "Capture on Fail".

**COMMAND SYNTAX**

:MTESt:FUNCtion:COF <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:MTESt:FUNCtion:COF?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the Capture on Fail and saves the screenshot to the U disk path "SIGLENT/".

Command message:

*:MTESt:FUNCtion:COF ON*

*MTESt:FUNC:COF ON*

Query message:

*MTESt:FUNC:COF?*

Response message:

*ON*

**:MTESt:FUNCtion:FTH****Command/Query****DESCRIPTION**

This command sets the state of the mask test function "Failure to History".

This command query returns the status of "Failure to History".

**COMMAND SYNTAX**

:MTESt:FUNCtion:FTH <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt:FUNCtion:FTH?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables Failure to History.

Command message:

*:MTESt:FUNCtion:FTH ON*

*MTESt:FUNC:FTH ON*

Query message:

*MTESt:FUNC:FTH?*

Response message:

*ON*

**RELATED COMMANDS**

:MTESt:OPERate

**:MTESt:FUNCtion:SOF****Command/Query****DESCRIPTION**

This command sets the state of the mask test function “Stop-on-Fail”.

This command query returns the status of “Stop-on-Fail”.

**COMMAND SYNTAX**

:MTESt:FUNCtion:SOF <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt:FUNCtion:SOF?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables Stop-on-Fail.

Command message:

*:MTESt:FUNCtion:SOF ON*

*MTESt:FUNC:SOF ON*

Query message:

*MTESt:FUNC:SOF?*

Response message:

*ON*

**:MTESt:IDISplay****Command/Query****DESCRIPTION**

This command sets the state of the mask test result display.

This command query returns the status of the mask test result display.

**COMMAND SYNTAX**

:MTESt:IDISplay <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt:IDISplay?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the display of the mask test.

Command message:

```
:MTESt:IDISplay ON
MTESt:IDIS ON
```

Query message:

```
MTESt:IDIS?
```

Response message:

```
ON
```

**RELATED COMMANDS**

:MTESt:COUNT

**:MTESt:MASK:CREate****Command****DESCRIPTION**

This command sets the mask X and mask Y of mask test.

**COMMAND SYNTAX**

:MTESt:MASK:CREate <XMARgin>,<YMARgin>

<XMARgin>:= Value in NR2 format. The range of the value is [0.08, 4.00]

<YMARgin>:= Value in NR2 format. The range of the value is [0.08, 4.00]

**EXAMPLE**

The following command sets the mask X to 0.8, the mask Y to 0.08.

Command message:

```
:MTESt:MASK:CREate 0.8,0.08
MTESt:MASK:CRE 0.8,0.08
```

**:MTEST:MASK:LOAD****Command****DESCRIPTION**

The command recalls the mask from internal or external memory locations.

**COMMAND SYNTAX**

:MTEST:MASK:LOAD <location>

<location>:= {INTernal,<num>|EXTernal,<path>}

- ◆ INTernal: the previous version supported saving masks internally by index. You can load these masks that has been saved in a previous version by specifying the parameter as INTernal.  
<num>:= {1|2|3|4}
- ◆ EXTernal: load the mask by specifying <path>.  
<path>:= Quoted string of path name with an extension ".msk" or ".smk". Users can recall from local, net storage or U-disk according to requirements.

| Path type   | Such as  |
|-------------|--|
| local       | "local/SIGLENT/test.msk"                                 |
| net storage | "net_storage/SIGLENT/test.smk"                           |
| U-disk      | "U-disk0/SIGLENT/test.msk"<br>"U-disk1/SIGLENT/test.smk" |

**Note:**

The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command recalls the mask from internal 1 which file name is SDS0001.msk.

Command message:

```
:MTEST:MASK:LOAD INTernal,1
```

```
MTES:MASK:LOAD INT,1
```

```
:MTEST:MASK:LOAD EXTernal,"local/SIGLENT/SDS0001.msk"
```

```
MTES:MASK:LOAD EXT,"local/SIGLENT/SDS0001.msk"
```

The following command recalls the mask from U-disk0 named "TEST.msk".

Command message:

```
MTES:MASK:LOAD EXTernal,"SIGLENT/TEST.msk"
```

**:MTESt:OPERate****Command/Query****DESCRIPTION**

This command sets the state of the mask test operation.

This command query returns the status of the mask test operation.

**COMMAND SYNTAX**

:MTESt:OPERate <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:MTESt:OPERate?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the operation of the mask test.

Command message:  
:MTESt:OPERate ON  
MTESt:OPER ON

Query message:  
MTESt:OPER?

Response message:  
ON

**:MTESt:RESet****Command****DESCRIPTION**

This command resets the mask test.

**COMMAND SYNTAX**

:MTESt:RESet

**EXAMPLE**

The following command resets the mask test.

Command message:  
:MTESt:RESet  
MTESt:RES

**RELATED COMMANDS**

:MTESt:OPERate



**:MTESt:SOURce**

**Command/Query**

**DESCRIPTION**

This command specifies the source of the mask test.

The query returns the current source of the mask test.

**COMMAND SYNTAX**

:MTESt:SOURce <source>

<source>:= {C<n>|Z<n>}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**Note:**

Only Z<n> can be selected when Zoom is on.

**QUERY SYNTAX**

:MTESt:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|Z<n>}

**EXAMPLE**

The following command sets the source of the mask test source as C1.

Command message:

```
:MTESt:SOURce C1
MTESt:SOUR C1
```

Query message:

```
MTESt:SOUR?
```

Response message:

```
C1
```

**:MTEST:TYPE****Command/Query****DESCRIPTION**

This command specifies the type of mask test.

The query returns the current type of mask test.

**COMMAND SYNTAX**

:MTEST:TYPE <type>

<type>:= {ALL\_IN|ALL\_OUT|ANY\_IN|ANY\_OUT}

- ◆ ALL\_IN means that all of the waveform elements must fall within the mask area.
- ◆ ALL\_OUT means that all of the waveform elements are all outside of the mask area.
- ◆ ANY\_IN means that the waveform is partially within the mask area.
- ◆ ANY\_OUT means that the waveform is partially outside the mask area.

**QUERY SYNTAX**

:MTEST:TYPE

**RESPONSE FORMAT**

<type>

<type>:= {ALL\_IN|ALL\_OUT|ANY\_IN|ANY\_OUT}

**EXAMPLE**

The following command sets the type of the mask test source as all in.

Command message:

```
:MTEST:TYPE ALL_IN
MTEST:TYPE ALL_IN
```

Query message:

```
MTEST:TYPE?
```

Response message:

```
ALL_IN
```

## RECall Commands

The :RECall subsystem commands control the recall of setups or waveform data to the oscilloscope.

- ◆ **:RECall:FDEFault**
- ◆ **:RECall:REFerence**
- ◆ **:RECall:SERase**
- ◆ **:RECall:SETup**

**:RECall:FDEFault****Command****DESCRIPTION**

This command recalls the factory settings.

**COMMAND SYNTAX**

:RECall:FDEFault

**EXAMPLE**

The following command recalls the factory settings.

Command message:

*:RECall:FDEFault*

*REC:FDEF*

**RELATED COMMANDS**

:RECall:SETup

**:RECall:REFeRence**

**Command**

**DESCRIPTION**

This command recalls the specified waveform file from an external USB memory device and copies it to the selected reference waveform.

**COMMAND SYNTAX**

:RECall:REFeRence <location>,<path>

<location>:= {REF<r>}

- ◆ REF denotes a reference waveform.

<r>:= {A|B|C|D}

<path>:= Quoted string of path with an extension ".ref"

Users can recall from local, net storage or U-disk according to requirements.

| Path type   | Such as  |
|-------------|--|
| local       | "local/SIGLENT/test.ref"                                 |
| net storage | "net_storage/SIGLENT/test.ref"                           |
| U-disk      | "U-disk0/SIGLENT/test.ref"<br>"U-disk1/SIGLENT/test.ref" |

**Note:**

The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command recalls the waveform "SIGLENT/math.ref" from an external U-disk and applies it to REFD.

Command message:

```
:RECall:REFeRence REFD,"U-disk0/SIGLENT/math.ref"
REC:REF REFD,"U-disk0/SIGLENT/math.ref"
```

**RELATED COMMANDS**

:SAVE:REFeRence

**:RECall:SERase****Command****DESCRIPTION**

This command deletes user defined files stored inside the oscilloscope, includes reference waveforms, internal setups, internal mask files, custom default setups, the waveform files copied from analog trace to AWG.

**COMMAND SYNTAX**

:RECall:SERase

**EXAMPLE**

The following command deletes user defined files stored inside the oscilloscope.

Command message:

*:RECall:SERase*

*REC:SER*

**:RECall:SETup**

**Command**

**DESCRIPTION**

This command will recall the saved settings file from internal or external sources.

**COMMAND SYNTAX**

:RECall:SETup <state>

<state>:= {INTernal,<num>|EXTernal,<path>}

<num>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1,10].

<path>:= Quoted string of path with an extension ".xml". Users can recall from local, net storage or U-disk according to requirements.

| Path type   | Such as  |
|-------------|--|
| local       | "local/SIGLENT/default.xml"                                    |
| net storage | "net_storage/SIGLENT/default.xml"                              |
| U-disk      | "U-disk0/SIGLENT/default.xml"<br>"U-disk1/SIGLENT/default.xml" |

**Note:**

- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.
- If the storage path type is not specified, it is recall from U-disk0 by default

**EXAMPLE**

The following command recalls the settings from internal file "SDS00001.xml".

Command message:

```
:RECall:SETup INTernal,1
REC:SET INT,1
```

The following command recalls the settings from the external file "SIGLENT/default.xml".

Command message:

```
:RECall:SETup EXTernal,"U-disk0/SIGLENT/default.xml"
REC:SET EXT,"SIGLENT/default.xml"
```

**RELATED COMMANDS**

```
:RECall:FDEFault
:SAVE:SETup
```

## REF Commands

The :REF<r> subsystem commands control the reference waveforms.

- ◆ :REF<r>:LABel
- ◆ :REF<r>:LABel:TEXT
- ◆ :REF<r>:DATA
- ◆ :REF<r>:DATA:SOURce
- ◆ :REF<r>:DATA:SCALe
- ◆ :REF<r>:DATA:POSition



**:REF<r>:LABel****Command/Query****DESCRIPTION**

The command is to turn the specified reference label on or off.

The query returns the state of the label associated with the specified reference.

**COMMAND SYNTAX**

:REF<r>:LABel <state>

<r>:= {A|B|C|D}

<state>:= {ON|OFF}

**QUERY SYNTAX**

:REF<r>:LABel?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the label display.

Command message:

*:REFA:LABel ON*

*REFA:LAB ON*

Query message:

*REFA:LAB?*

Response message:

*ON*

**RELATED COMMANDS**

:REF<r>:LABel:TEXT

**:REF<r>:LABel:TEXT****Command/Query****DESCRIPTION**

The command sets the selected REF label to the string that follows. Setting a label for a REF also adds the name to the label list in non-volatile memory (replacing the oldest label in the list).

The query returns the current label text of the selected reference waveform.

**COMMAND SYNTAX**

:REF<r>:LABel:TEXT <string>

<r>:= {A|B|C|D}

<string>:= Quoted string of ASCII text. The length of the string is limited to 20 characters.

**QUERY SYNTAX**

:REF<r>:LABel:TEXT?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the reference waveform label text to REFA.

Command message:

*:REFA:LABel:TEXT "REFA"*

*REFA:LAB:TEXT "REFA"*

Query message:

*REFA:LAB:TEXT?*

Response message:

*"REFA"*

**RELATED COMMANDS**

:REF<r>:LABel

**:REF<r>:DATA****Command****DESCRIPTION**

The command controls the display and saving of reference waveforms.

**COMMAND SYNTAX**

:REF<r>:DATA <operation>

<r>:= {A|B|C|D}

<operation>:= {LOAD|UNLoad|SAVE,<source>}

- ◆ LOAD means to call up the reference waveform display.
- ◆ UNLoad means to turn off the reference waveform display.
- ◆ SAVE means to save the waveform to the reference waveform.

<source>:= {C<n>|F<x>|D<d>}

- ◆ C denotes an analog channel.
- ◆ F denotes a math function.
- ◆ D denotes a digital channel.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command turns on REFA.

Command message:

*:REFA:DATA LOAD*

*REFA:DATA LOAD*

**:REF<r>:DATA:SOURce**

### Query

#### DESCRIPTION

This query returns the source of the current reference channel.

#### QUERY SYNTAX

:REF<r>:DATA:SOURce?

<r>:= {A|B|C|D}

#### RESPONSE FORMAT

<source>

<source>:= {C<n>|F<x>|D<d>}

- ◆ C denotes an analog channel.
- ◆ F denotes a math function.
- ◆ D denotes a digital channel.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

#### EXAMPLE

The following query returns the source of REFA.

Query message:

*REFA:DATA:SOUR?*

Response message:

*C1*

**:REF<r>:DATA:SCALe**

**Command/Query**

**DESCRIPTION**

The command sets the vertical scale of the current reference channel. This command is only used when the current reference channel has been stored, and the display state is on.

The query returns the vertical scale of the current reference channel.

**COMMAND SYNTAX**

:REF<r>:DATA:SCALe <value>

<r>:= {A|B|C|D}

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The scale range of the reference waveform is the same as that of the reference source.

**QUERY SYNTAX**

:REF<r>:DATA:SCALe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

When the Reference function is on, and REFA has been saved, the following command sets the vertical scale of REFA to 100 mV.

Command message:

*:REFA:DATA:SCALe 1.00E-01*

*REFA:DATA:SCAL 1.00E-01*

Query message:

*REFA:DATA:SCAL?*

Response message:

*1.00E-01*

**RELATED COMMANDS**

:REF<r>:DATA:POSition

**:REF<r>:DATA:POSition****Command/Query****DESCRIPTION**

The command sets the vertical offset of the current reference channel. This command is only used when the current reference channel has been saved, and the display state is on.

This query returns the vertical offset of the current reference channel.

**COMMAND SYNTAX**

:REF<r>:DATA:POSition <value>

<r>:= {A|B|C|D}

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The position range of the reference waveform is the same as that of the reference source.

**QUERY SYNTAX**

:REF<r>:DATA:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

When the Reference function is on, REFA has been saved and the scale is 2 V, the following command sets the current reference channel vertical offset to 0.2 V.

Command message:

*:REFA:DATA:POSition 2.00E-01*

*REFA:DATA:POS 2.00E-01*

Query message:

*REFA:DATA:POS?*

Response message:

*2.00E-01*

**RELATED COMMANDS**

:REF<r>:DATA:SCALE

## SAVE Commands

The SAVE subsystem commands control to save oscilloscope setups and waveform data to internal or external memory locations.

- ◆ **:SAVE:BINary**
- ◆ **:SAVE:CSV**
- ◆ **:SAVE:DEFault**
- ◆ **:SAVE:IMAGe**
- ◆ **:SAVE:MATLab**
- ◆ **:SAVE:REFerence**
- ◆ **:SAVE:SETup**

**:SAVE:BINary**

**Command**

**DESCRIPTION**

This command saves the binary data of the channel displayed on the screen to an external USB memory device.

**COMMAND SYNTAX**

:SAVE:BINary <path>,<src>

<path>:= Quoted string of path with an extension “.bin”  
Users can save to local, net storage or U-disk according to requirements

| Path type   | Such as                  |
|-------------|--------------------------|
| local       | “local/SIGLENT/test.bin” |
| net storage | “net_storage/test.bin”   |
| U-disk      | “U-disk0/test.bin”       |

<src>:= {C<n>|Z<n>|F<x>|M<m>|D0\_D15|ZD0\_ZD15}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ D0\_D15 denotes a digital waveform. Data display by bit.
- ◆ ZD0\_ZD15 denotes a zoomed digital waveform. Data display by bit.

**Note:**

- When save to internal, the default path is local.
- When save to external, if the path type is not set, it is stored to u-disk0 by default
- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.
- If the parameter <src> is not specified, the command is invalid.

**EXAMPLE**

Here is an example of saving a file to an external drive when channel 1 is enabled. The following command will save their waveform data to the external file "c1.bin".

Command message:

```
:SAVE:BINary "U-disk0/Siglent/c1.bin",C1
SAVE:BIN "U-disk0/Siglent/c1.bin",C1
```



**:SAVE:CSV****Command****DESCRIPTION**

This command saves the waveform data of the specified channel to an external U disk/USB memory device in CSV format.

**COMMAND SYNTAX**

**:SAVE:CSV <path>,<source>,<state>**

<path>:= Quoted string of path with an extension “.csv”. Users can save to local, net storage or U-disk according to requirements

| Path type   | Such as                         |
|-------------|---------------------------------|
| local       | <i>“local/SIGLENT/test.csv”</i> |
| net storage | <i>“net_storage/test.csv”</i>   |
| U-disk      | <i>“U-disk0/test.csv”</i>       |

<source>:=

{C<n>|Z<n>|F<x>|M<m>|D0\_D15|DIGital|ZD0\_ZD15|ZDIGital}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ D0\_D15 denotes a digital waveform. Data display by bit.
- ◆ DIGital denotes a digital waveform. Data display by bus.
- ◆ ZD0\_ZD15 denotes a zoomed digital waveform. Data display by bit.
- ◆ ZDIGital denotes a zoomed digital waveform. Data display by bus.

<state>:= {OFF|ON}

- ◆ ON enables parameter save. This adds vertical scale values, horizontal timebase settings, and more instrument configuration information to the file.
- ◆ OFF means to disables parameter save.

**Note:**

- When save to internal, the default path is local.
- When save to external, if the path type is not set, it is stored to u-disk0 by default
- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command saves data and parameters of channel 1 to the local file “local/SIGLENT/channel1.csv”.

Command message:

*:SAVE:CSV "local/SIGLENT/channel1.csv",C1,ON*

**:SAVE:DEFault****Command****DESCRIPTION**

This command saves the current settings or factory settings as default settings.

**COMMAND SYNTAX**

:SAVE:DEFault <set>

<set>:= {CUSTom|FACTory}

- ◆ CUSTom means the current settings.
- ◆ FACTory means factory settings.

**EXAMPLE**

The following command saves the current settings to default settings.

Command message:

*:SAVE:DEFault CUSTom*

*SAVE:DEF CUST*

**RELATED COMMANDS**

:RECall:SETup

**:SAVE:IMAGe****Command****DESCRIPTION**

This command saves the screenshot to external storage.

**COMMAND SYNTAX**

**:SAVE:IMAGe** <path>,<type>,<invert>[,<menu>]

<path>:= Quoted string of path with an extension “.bmp” or “.jpg” or “.png”.

Users can save to local, net storage or U-disk according to requirements

| Path type   | Such as                         |
|-------------|---------------------------------|
| local       | <i>“local/SIGLENT/test.bmp”</i> |
| net storage | <i>“net_storage/test.jpg”</i>   |
| U-disk      | <i>“U-disk0/test.png”</i>       |

<type>:= {BMP|JPG|PNG}

<invert>:= {OFF|ON}

- ◆ ON will store images that have inverted colors. This means that a normally black background will be white when inverted. This setting is recommended if you plan on printing the image as an inverted image with a white background will save on ink.
- ◆ OFF will store images that are identical to the display of the instrument.

<menu>:= {MOFf|MON}

- ◆ MON will store images that include all menu information bars
- ◆ MOFf: images do not include the top menu, right menu, and bottom right time information bar

**Note:**

- When save to internal, the default path is local.
- When save to external, if the path type is not set, it is stored to u-disk0 by default
- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command saves the screenshot in BMP format to the external file “U-disk0/SIGLENT/screen.bmp” .

Command message:

*:SAVE:IMAGe "U-disk0/SIGLENT/screen.bmp",BMP,ON*  
*SAVE:IMAG "U-disk0/SIGLENT/screen.bmp",BMP,ON*

**RELATED COMMANDS**

**:PRINT**

**:SAVE:MATLab****Command****DESCRIPTION**

This command saves the waveform data of the specified channel to an external USB memory device in Matlab format.

**COMMAND SYNTAX**

`:SAVE:MATLab <path>,<source>`

`<path>`:= Quoted string of path with an extension “.mat”. Users can save to local, net storage or U-disk according to requirements

| Path type   | Such as                                |
|-------------|--|
| local       | <code>"/local/SIGLENT/test.mat"</code> |
| net storage | <code>"net_storage/test.mat"</code>    |
| U-disk      | <code>"U-disk0/test.mat"</code>        |

`<source>`:= {C<n>|Z<n>|F<x>|M<m>|D0\_D15|DIGital|ZD0\_ZD15|ZDIGital}

- ◆ C denotes an analog channel.
- ◆ Z denotes a zoomed source.
- ◆ F denotes a math function.
- ◆ M denotes a memory waveform
- ◆ D0\_D15 denotes a digital waveform. Data display by bit.
- ◆ DIGital denotes a digital waveform. Data display by bus.
- ◆ ZD0\_ZD15 denotes a zoomed digital waveform. Data display by bit.
- ◆ ZDIGital denotes a zoomed digital waveform. Data display by bus.

**Note:**

- When save to internal, the default path is local.
- When save to external, if the path type is not set, it is stored to u-disk0 by default
- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command saves data of channel 1 to the external file “U-disk0/SIGLENT/channel1.mat”.

Command message:

```
:SAVE:MATLab "U-disk0/SIGLENT/channel.mat",C1
SAVE:MATL "U-disk0/SIGLENT/channel.mat",C1
```

**:SAVE:REFerence**

**Command**

**DESCRIPTION**

This command saves the selected channel waveform to external memory as reference.

**COMMAND SYNTAX**

:SAVE:REFerence <path>,<source>

<path>:= Quoted string of path with an extension “.ref”. Users can save to local, net storage or U-disk according to requirements

| Path type   | Such as                         |
|-------------|---------------------------------|
| local       | <i>“local/SIGLENT/test.ref”</i> |
| net storage | <i>“net_storage/test.ref”</i>   |
| U-disk      | <i>“U-disk0/test.ref”</i>       |

<source>:= {C<n>|F<x>|D<d>}

- ◆ C denotes an analog channel.
- ◆ F denotes a math function.
- ◆ D denotes a digital channel.

**Note:**

The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

The following command saves the waveform of channel 1 as a reference to the local file “local/SIGLENT/channel.ref”.

Command message:

*:SAVE:REFerence "local/SIGLENT/channel.ref",C1  
SAVE:REF "local/SIGLENT/channel.ref",C1*

**RELATED COMMANDS**

:RECall:REFerence

**:SAVE:SETup**

**Command**

**DESCRIPTION**

This command saves the current settings to internal or external memory locations.

**COMMAND SYNTAX**

:SAVE:SETup <setup\_num>

<setup\_num>:= {INTernal,<num>|EXTernal,<path>}

<num>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 10].

<path>:= Quoted string of path with an extension “.xml”. Users can recall from local,net storage or U-disk according to requirements

| Path type   | Such as                           |
|-------------|-----------------------------------|
| local       | “local/SIGLENT/default.xml”       |
| net storage | “net_storage/SIGLENT/default.xml” |
| U-disk      | “U-disk0/SIGLENT/default.xml”     |

**Note:**

- When save to internal, the default path is local. And the setup file will be stored in local with the name "SDS000x.xml"
- When save to external, if the path type is not set, it is stored to u-disk0 by default
- The file format is not automatically determined by the file name extension. You need to choose a file name with an extension which is consistent with the selected file format.

**EXAMPLE**

There are two ways to save the current settings to internal file “SDS00001.xml”.

Command message:

*:SAVE:SETup INTernal,1*  
*SAVE:SET INT,1*

*:SAVE:SETup EXTernal,"local/SDS00001.xml"*  
*:SAVE:SET EXT,"local/SDS00001.xml"*

**RELATED COMMANDS**

:SAVE:DEFault  
 :RECall:SETup

## **SEARch Commands**

The :SEARch subsystem commands control the search functions of the oscilloscope.

- ◆ **:SEARch**
- ◆ **:SEARch:MODE**
- ◆ **:SEARch:COUNT**
- ◆ **:SEARch:EVENT**
- ◆ **:SEARch:COPY**
- ◆ **:SEARch:EDGE Commands**
- ◆ **:SEARch:SLOPe Commands**
- ◆ **:SEARch:PULSe Commands**
- ◆ **:SEARch:INTerval Commands**
- ◆ **:SEARch:RUNT Commands**

**:SEARCh****Command/Query****DESCRIPTION**

The command sets the switch of the search function.

This query returns the current status of the search function.

**COMMAND SYNTAX**

:SEARCh <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SEARCh?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the search function.

Command message:

*:SEARCh ON*  
*SEAR ON*

Query message:

*SEAR?*

Response message:

*ON*



**:SEARch:MODE****Command/Query****DESCRIPTION**

The command sets the mode of search.

The query returns the current mode of search.

**COMMAND SYNTAX**

:SEARch:MODE <mode>

<mode>:= {EDGE|SLOPe|PULSE|INTerval|RUNT}

**QUERY SYNTAX**

:SEARch:MODE?

**RESPONSE FORMAT**

<mode>

<mode>:= {EDGE|SLOPe|PULSE|INTerval|RUNT}

**EXAMPLE**

The following command sets the mode of search to edge search.

Command message:

*:SEARch:MODE EDGE*

*SEAR:MODE EDGE*

Query message:

*SEAR:MODE?*

Command message:

*EDGE*

**:SEARch:COUNt****Query****DESCRIPTION**

The query returns the total number of search events in the current screen.

**QUERY SYNTAX**

:SEARch:COUNt?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following query returns the number of search events in the current screen.

Query message:

*SEARch:COUNt?*

*SEAR:COUN?*

Response message:

*10*

**:SEARCh:EVENT****Query****DESCRIPTION**

The query returns the index of the search event in the center of the screen when the oscilloscope acquisition is stopped.

**QUERY SYNTAX**

:SEARCh:EVENT?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following query returns the central event index.

Query message:

*SEARCh:EVENT?*

*SEAR:EVENT?*

Response message:

*5*

**:SEARCh:COPY****Command****DESCRIPTION**

The command synchronizes the search settings with the trigger settings.

**COMMAND SYNTAX**

:SEARCh:COPY <operation>

<operation>:= {FROMtrigger|TOTRigger|CANCel}

- ◆ FROMtrigger means copy trigger settings to search.
- ◆ TOTRigger means copy search settings to trigger.
- ◆ CANCEL can undo the above two copying operations.

**EXAMPLE**

The following command copies the trigger settings to search.

Command message:

*:SEARCh:COPY FROMtrigger*

*SEAR:COPY FROM*

## **:SEARch:EDGE Commands**

The :SEARch:EDGE subsystem commands control the edge search parameters.

- ◆ **:SEARch:EDGE:SOURce**
- ◆ **:SEARch:EDGE:SLOPe**
- ◆ **:SEARch:EDGE:LEVel**

**:SEARch:EDGE:SOURce**

**Command/Query**

**DESCRIPTION**

The command sets the search source of the edge search.

The query returns the current search source of the edge search.

**COMMAND SYNTAX**

:SEARch:EDGE:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:SEARch:EDGE:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the search source of the edge search as C1.

Command message:

*:SEARch:EDGE:SOURce C1*  
*SEAR:EDGE:SOUR C1*

Query message:

*SEAR:EDGE:SOUR?*

Response message:

*C1*

**RELATED COMMANDS**

:SEARch:EDGE:LEVel

**:SEARch:EDGE:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the edge search.

The query returns the current slope setting of the edge search.

**COMMAND SYNTAX**

:SEARch:EDGE:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing|ALternate}

**QUERY SYNTAX**

:SEARch:EDGE:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing|ALternate}

**EXAMPLE**

The following command set the rising slope to the edge search.

Command message:

```
:SEARch:EDGE:SLOPe RISing  
SEAR:EDGE:SLOP RIS
```

Query message:

```
SEAR:EDGE:SLOP?
```

Response message:

```
RISing
```

**:SEARch:EDGE:LEVel****Command/Query****DESCRIPTION**

The command sets the search level of the edge search.

The query returns the current search level value of the edge search.

**COMMAND SYNTAX**

:SEARch:EDGE:LEVel <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:SEARch:EDGE:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the search level of the edge search to 0.5 V.

Command message:

```
:SEARch:EDGE:LEVel 5.00E-01  
SEAR:EDGE:LEV 5.00E-01
```

Query message:

```
SEAR:EDGE:LEV?
```

Response message:

```
5.00E-01
```

**RELATED COMMANDS**

:SEARch:EDGE:SOURce

## **:SEARch:SLOPe Commands**

The :SEARch:SLOPe subsystem commands control the slope search parameters.

- ◆ **:SEARch:SLOPe:SOURce**
- ◆ **:SEARch:SLOPe:SLOPe**
- ◆ **:SEARch:SLOPe:HLEVel**
- ◆ **:SEARch:SLOPe:LLEVel**
- ◆ **:SEARch:SLOPe:LIMit**
- ◆ **:SEARch:SLOPe:TUPPer**
- ◆ **:SEARch:SLOPe:TLOWer**

**:SEARch:SLOPe:SOURce****Command/Query****DESCRIPTION**

The command sets the search source of the slope search.

The query returns the current search source of the slope search.

**COMMAND SYNTAX**

:SEARch:SLOPe:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:SEARch:SLOPe:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the search source of the slope search to C2 (channel 2).

Command message:

*:SEARch:SLOPe:SOURce C2*

*SEAR:SLOP:SOUR C2*

Query message:

*SEAR:SLOP:SOUR?*

Response message:

*C2*



**:SEARch:SLOPe:SLOPe**  
**Command/Query****DESCRIPTION**

The command sets the slope of the slope search.

The query returns the current slope of the slope search.

**COMMAND SYNTAX**

:SEARch:SLOPe:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:SEARch:SLOPe:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the slope search.

Command message:

*:SEARch:SLOPe:SLOPe RISing*  
*SEAR:SLOP:SLOP RIS*

Query message:

*SEAR:SLOP:SLOP?*

Response message:

*RISing*

**:SEARCh:SLOPe:HLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the high level of the slope search.

The query returns the current high level of the slope search.

**COMMAND SYNTAX**

:SEARCh:SLOPe:HLEVel <high\_level\_value>

<high\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The high level value cannot be less than the low level value using by the command :SEARCh:SLOPe:LLEVel.

**QUERY SYNTAX**

:SEARCh:SLOPe:HLEVel?

**RESPONSE FORMAT**

<high\_level\_value>

<high\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the high level of the slope search to 0.5 V.

Command message:

*:SEARCh:SLOPe:HLEVel 5.00E-01*  
*SEAR:SLOP:HLEV 5.00E-01*

Query message:

*SEAR:SLOP:HLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:SEARCh:SLOPe:LLEVel

**:SEARCh:SLOPe:LLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the low level of the slope search.

The query returns the current low level of the slope search.

**COMMAND SYNTAX**

:SEARCh:SLOPe:LLEVel <low\_level\_value>

<low\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The low level value cannot be greater than the low level value using by the command :SEARCh:SLOPe:HLEVel.

**QUERY SYNTAX**

:SEARCh:SLOPe:LLEVel?

**RESPONSE FORMAT**

<low\_level\_value>

<low\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the low level of the slope search to -0.5 V.

Command message:

*:SEARCh:SLOPe:LLEVel -5.00E-01*  
*SEAR:SLOP:LLEV -5.00E-01*

Query message:

*SEAR:SLOP:LLEV?*

Response message:

*-5.00E-01*

**RELATED COMMANDS**

:SEARCh:SLOPe:HLEVel

**:SEARch:SLOPe:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the slope search.

The query returns the current limit range type of the slope search.

**COMMAND SYNTAX**

:SEARch:SLOPe:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:SEARch:SLOPe:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the slope search to LESSthan.

Command message:

```
:SEARch:SLOPe:LIMit LESSthan  
SEAR:SLOP:LIM LESS
```

Query message:

```
SEAR:SLOP:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

```
:SEARch:SLOPe:TUPPer  
:SEARch:SLOPe:TLOWer
```

**:SEARch:SLOPe:TUPPer**  
**Command/Query**

**DESCRIPTION**

The command sets the upper value of the slope search limit type.

The query returns the current upper value of the slope search limit type.

**COMMAND SYNTAX**

:SEARch:SLOPe:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :SEARch:SLOPe:TLOWer.
- The command is not valid when the limit range type is GREATERthan.

**QUERY SYNTAX**

:SEARch:SLOPe:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper value of the slope search to 30 ns, when the limit range type is OUTER.

Command message:

*:SEARch:SLOPe:TUPPer 3.00E-08*

*SEAR:SLOP:TUPP 3.00E-08*

Query message:

*SEAR:SLOP:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:SEARch:SLOPe:LIMit

:SEARch:SLOPe:TLOWer

**:SEARch:SLOPe:TLOWer**  
**Command/Query**

**DESCRIPTION**

The command sets the lower value of the slope search limit type.

The query returns the current lower value of the slope search limit type.

**COMMAND SYNTAX**

:SEARch:SLOPe:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :SEARch:SLOPe:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:SEARch:SLOPe:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the slope search to 10 ns.

Command message:

*:SEARch:SLOPe:TLOWer 1.00E-08*  
*SEAR:SLOP:TLOW 1.00E-08*

Query message:

*SEAR:SLOP:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:SEARch:SLOPe:LIMit  
 :SEARch:SLOPe:TUPPer

## **:SEARch:PULSe Commands**

The :SEARch:PULSe subsystem commands control the pulse search parameters.

- ◆ **:SEARch:PULSe:SOURce**
- ◆ **:SEARch:PULSe:POLarity**
- ◆ **:SEARch:PULSe:LEVel**
- ◆ **:SEARch:PULSe:LIMit**
- ◆ **:SEARch:PULSe:TUPPer**
- ◆ **:SEARch:PULSe:TLOWer**

**:SEARch:PULSe:SOURce****Command/Query****DESCRIPTION**

The command sets the search source of the pulse search.

The query returns the current search source of the pulse search.

**COMMAND SYNTAX**

:SEARch:PULSe:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:SEARch:PULSe:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the polarity of the pulse search as channel 2.

Command message:

```
:SEARch:PULSe:SOURce C2  
SEAR:PULS:SOUR C2
```

Query message:

```
SEAR:PULS:SOUR?
```

Response message:

```
C2
```



**:SEARch:PULSe:POLarity****Command/Query****DESCRIPTION**

The command sets the polarity of the pulse search.

The query returns the current polarity of the pulse search.

**COMMAND SYNTAX**

:SEARch:PULSe:POLarity <polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**QUERY SYNTAX**

:SEARch:PULSe:POLarity?

**RESPONSE FORMAT**

<polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**EXAMPLE**

The following command sets the polarity of the pulse search to POSitive.

Command message:

```
:SEARch:PULSe:POLarity POSitive  
SEAR:PULS:POL POS
```

Query message:

```
SEAR:PULS:POL?
```

Response message:

```
POSitive
```

**:SEARch:PULSe:LEVel**  
**Command/Query**

**DESCRIPTION**

The command sets the search level of the pulse search.

The query returns the current search level of the pulse search.

**COMMAND SYNTAX**

:SEARch:PULSe:LEVel <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:SEARch:PULSe:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the search level of the pulse search to 0.5 V.

Command message:

*:SEARch:PULSe:LEVel 5.00E-01*  
*SEAR:PULS:LEV 5.00E-01*

Query message:

*SEAR:PULS:LEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:SEARch:PULSe:SOURce

**:SEARch:PULSe:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the pulse search.

The query returns the current limit range type of the pulse search.

**COMMAND SYNTAX**

:SEARch:PULSe:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:SEARch:PULSe:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the search limit of the pulse search to inner.

Command message:

```
:SEARch:PULSe:LIMit INNER  
SEAR:PULS:LIM INN
```

Query message:

```
SEAR:PULS:LIM?
```

Response message:

```
INNER
```

**RELATED COMMANDS**

:SEARch:PULSe:TUPPer

:SEARch:PULSe:TLOWer

**:SEARch:PULSe:TUPPer**

**Command/Query**

**DESCRIPTION**

The command sets the upper value of the pulse search limit type.

The query returns the current upper value of the pulse search limit type.

**COMMAND SYNTAX**

:SEARch:PULSe:TUPPer <value>

<value>:= Value in NR3 format.The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :SEARch:PULSe:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:SEARch:PULSe:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the upper time of the pulse search to 30 ns.

Command message:

*:SEARch:PULSe:TUPPer 3.00E-08*

*SEAR:PULS:TUPP 3.00E-08*

Query message:

*SEAR:PULS:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:SEARch:PULSe:LIMit

:SEARch:PULSe:TLOWer

**:SEARch:PULSe:TLOWer****Command/Query****DESCRIPTION**

The command sets the lower value of the pulse search limit type.

The query returns the current lower value of the pulse search limit type.

**COMMAND SYNTAX**

:SEARch:PULSe:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :SEARch:PULSe:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:SEARch:PULSe:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the pulse search to 10 ns.

Command message:

*:SEARch:PULSe:TLOWer 1.00E-08*

*SEAR:PULS:TLOW 1.00E-08*

Query message:

*SEAR:PULS:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:SEARch:PULSe:LIMit

:SEARch:PULSe:TUPPer

**:SEARch:INTerval Commands**

The :SEARch:INTerval subsystem commands control the interval search parameters.

- ◆ **:SEARch:INTerval:SOURce**
- ◆ **:SEARch:INTerval:SLOPe**
- ◆ **:SEARch:INTerval:LEVel**
- ◆ **:SEARch:INTerval:LIMit**
- ◆ **:SEARch:INTerval:TUPPer**
- ◆ **:SEARch:INTerval:TLOWer**

**:SEARCh:INTerval:SOURce****Command/Query****DESCRIPTION**

The command sets the search source of the interval search.

The query returns the current search source of the interval search.

**COMMAND SYNTAX**

:SEARCh:INTerval:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:SEARCh:INTerval:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the search source of the interval search as channel 1.

Command message:

*:SEARCh:INTerval:SOURce C1*

*SEAR:INT:SOUR C1*

Query message:

*SEAR:INT:SOUR?*

Response message:

*C1*

**:SEARch:INTerval:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the interval search.

The query returns the current slope of the interval search.

**COMMAND SYNTAX**

:SEARch:INTerval:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:SEARch:INTerval:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the interval search.

Command message:

*:SEARch:INTerval:SLOPe RISing*

*SEAR:INT:SLOP RIS*

Query message:

*SEAR:INT:SLOP?*

Response message:

*RISing*



**:SEARch:INTerval:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the search level of the interval search.

The query returns the current search level of the interval search.

**COMMAND SYNTAX**

:SEARch:INTerval:LEVel <level\_value>

<level\_value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:SEARch:INTerval:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format

**EXAMPLE**

The following command sets the search level of the interval search to 0.5 V.

Command message:

*:SEARch:INTerval:LEVel 5.00E-01*  
*SEARr:INT:LEV 5.00E-01*

Query message:

*SEAR:INT:LEV?*

Response message:

*5.00E-01*

**:SEARch:INTerval:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the interval search.

The query returns the current limit range type of the interval search.

**COMMAND SYNTAX**

:SEARch:INTerval:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:SEARch:INTerval:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the interval search to LESSthan.

Command message:

```
:SEARch:INTerval:LIMit LESSthan  
SEAR:INT:LIM LESS
```

Query message:

```
SEAR:INT:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:SEARch:INTerval:TUPPer

:SEARch:INTerval:TLOWer

**:SEARch:INTerval:TUPPer****Command/Query****DESCRIPTION**

The command sets the upper value of the interval search limit type.

The query returns the current upper value of the interval search limit type.

**COMMAND SYNTAX**

:SEARch:INTerval:TUPPer <value>

<value>:= Value in NR3 format. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :SEARch:INTerval:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:SEARch:INTerval:TUPPer?

**RESPONSE FORMAT**

<tupper\_value>

<tupper\_value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the time upper value of the interval search to 30 ns.

Command message:

*:SEARch:INTerval:TUPPer 3.00E-08*

*SEAR:INT:TUPP 3.00E-08*

Query message:

*SEAR:INT:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:SEARch:INTerval:LIMit

:SEARch:INTerval:TLOWer

**:SEARch:INTerval:TLOWer****Command/Query****DESCRIPTION**

The command sets the lower value of the interval search limit type.

The query returns the current lower value of the interval search limit type.

**COMMAND SYNTAX**

:SEARch:INTerval:TLOWer <value>

<value>:= Value in NR3 format. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :SEARch:INTerval:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:SEARch:INTerval:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the time lower value of the interval search to 10 ns.

Command message:

```
:SEARch:INTerval:TLOWer 1.00E-08
SEAR:INT:TLOW 1.00E-08
```

Query message:

```
SEAR:INT:TLOW?
```

Response message:

```
1.00E-08
```

**RELATED COMMANDS**

```
:SEARch:INTerval:LIMit
:SEARch:INTerval:TUPPer
```

## **:SEARch:RUNT Commands**

The :SEARch:RUNT subsystem commands control the runt search parameters.

- ◆ **:SEARch:RUNT:SOURce**
- ◆ **:SEARch:RUNT:POLarity**
- ◆ **:SEARch:RUNT:HLEVel**
- ◆ **:SEARch:RUNT:LLEVel**
- ◆ **:SEARch:RUNT:LIMit**
- ◆ **:SEARch:RUNT:TUPPer**
- ◆ **:SEARch:RUNT:TLOWer**

**:SEARch:RUNT:SOURce****Command/Query****DESCRIPTION**

The command sets the search source of the runt search.

The query returns the current search source of the runt search.

**COMMAND SYNTAX**

:SEARch:RUNT:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:SEARch:RUNT:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the search source of the runt search to channel 2

Command message:

*:SEARch:RUNT:SOURce C2*  
*SEAR:RUNT:SOUR C2*

Query message:

*SEAR:RUNT:SOUR?*

Response message:

*C2*

**:SEARch:RUNT:POLarity****Command/Query****DESCRIPTION**

The command sets the polarity of the runt search.

The query returns the current polarity of the runt search.

**COMMAND SYNTAX**

:SEARch:RUNT:POLarity <polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**QUERY SYNTAX**

:SEARch:RUNT:POLarity?

**RESPONSE FORMAT**

<polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**EXAMPLE**

The following command sets the polarity of the runt search to POSitive.

Command message:

```
:SEARch:RUNT:POLarity POSitive  
SEAR:RUNT:POL POS
```

Query message:

```
SEAR:RUNT:POL?
```

Response message:

```
POSitive
```

**:SEARCh:RUNT:HLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the high search level of the runt search.

The query returns the current high search level of the runt search.

**COMMAND SYNTAX**

:SEARCh:RUNT:HLEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The high level value cannot be less than the low level value using by the command :SEARCh:RUNT:LLEVel.

**QUERY SYNTAX**

:SEARCh:RUNT:HLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the high search level of the runt search to 0.5 V.

Command message:

*:SEARCh:RUNT:HLEVel 5.00E-01*  
*SEAR:RUNT:HLEV 5.00E-01*

Query message:

*SEAR:RUNT:HLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:SEARCh:RUNT:LLEVel



**:SEARch:RUNT:LLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the low search level of the runt search.

The query returns the current low search level of the runt search.

**COMMAND SYNTAX**

:SEARch:RUNT:LLEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The low level value cannot be greater than the high level value using by the command :SEARch:RUNT:HLEVel.

**QUERY SYNTAX**

:SEARch:RUNT:LLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the low search level of the runt search to -0.5 V.

Command message:

*:SEARch:RUNT:LLEVel - 5.00E-01*  
*SEAR:RUNT:LLEV - 5.00E-01*

Query message:

*SEAR:RUNT:LLEV?*

Response message:

*-5.00E-01*

**RELATED COMMANDS**

:SEARch:RUNT:HLEVel

**:SEARCh:RUNT:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the runt search.

The query returns the current limit range type of the runt search.

**COMMAND SYNTAX**

:SEARCh:RUNT:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:SEARCh:RUNT:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the runt search to LESSthan.

Command message:

```
:SEARCh:RUNT:LIMit LESSthan  
SEAR:RUNT:LIM LESS
```

Query message:

```
SEAR:RUNT:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

```
:SEARCh:RUNT:TUPPer  
:SEARCh:RUNT:TLOWer
```

**:SEARCh:RUNT:TUPPer**

**Command/Query**

**DESCRIPTION**

The command sets the upper value of the runt search limit type.

The query returns the current upper value of the runt search limit type.

**COMMAND SYNTAX**

:SEARCh:PULse:RUNT <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :SEARCh:RUNT:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:SEARCh:RUNT:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the upper time of the runt search to 30 ns.

Command message:

*:SEARCh:RUNT:TUPPer 3.00E-08*  
*SEAR:RUNT:TUPP 3.00E-08*

Query message:

*SEAR:RUNT:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:SEARCh:RUNT:LIMit  
:SEARCh:RUNT:TLOWer

**:SEARch:RUNT:TLOWer**  
**Command/Query**

**DESCRIPTION**

The command sets the lower value of the runt search limit type.

The query returns the current lower value of the runt search limit type.

**COMMAND SYNTAX**

:SEARch:RUNT:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :SEARch:RUNT:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:SEARch:RUNT:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the lower time of the runt search to 10 ns.

Command message:

*:SEARch:RUNT:TLOWer 1.00E-08*  
*SEAR:RUNT:TLOW 1.00E-08*

Query message:

*SEAR:RUNT:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:SEARch:RUNT:LIMit  
:SEARch:RUNT:TUPPer

## SYSTEM Commands

The :SYSTEM subsystem commands control the basic system functions of the oscilloscope.

- ◆ :SYSTEM:BUZZer
- ◆ :SYSTEM:CLOCK
- ◆ :SYSTEM:COMMunicate:LAN:GATeway
- ◆ :SYSTEM:COMMunicate:LAN:IPADdress
- ◆ :SYSTEM:COMMunicate:LAN:MAC
- ◆ :SYSTEM:COMMunicate:LAN:SMASK
- ◆ :SYSTEM:COMMunicate:LAN:TYPE
- ◆ :SYSTEM:COMMunicate:VNCPort
- ◆ :SYSTEM:DATE
- ◆ :SYSTEM:EDUMode
- ◆ :SYSTEM:MENU
- ◆ :SYSTEM:NStorage
- ◆ :SYSTEM:NStorage:CONNect
- ◆ :SYSTEM:NStorage:DISConnect
- ◆ :SYSTEM:NStorage:STATus
- ◆ :SYSTEM:PON
- ◆ :SYSTEM:REBoot
- ◆ :SYSTEM:REMOte
- ◆ :SYSTEM:SELFCal
- ◆ :SYSTEM:SHUTdown
- ◆ :SYSTEM:SSAVer
- ◆ :SYSTEM:TIME
- ◆ :SYSTEM:TOUCh

**:SYSTem:BUZZer****Command/Query****DESCRIPTION**

The command the status of the buzzer.

The query returns the current status of the buzzer.

**COMMAND SYNTAX**

:SYSTem:BUZZer <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SYSTem:BUZZer?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the oscilloscope buzzer.

Command message:  
:SYSTem:BUZZer ON  
SYST:BUZZ ON

Query message:  
SYST:BUZZ?

Response message:  
ON

**:SYSTem:CLOCK**

**Command/Query**

**DESCRIPTION**

The command sets the oscilloscope clock source and the state of the 10 MHz clock output.

The query returns the oscilloscope current clock source and the state of the 10 MHz clock output.

**COMMAND SYNTAX**

:SYSTem:CLOCK <source>

<source>:= {EXT|IN\_ON|IN\_OFF}

- ◆ EXT selects the external clock source. The 10 MHz output will be automatically disabled.
- ◆ IN\_ON selects the internal clock source and enables the 10 MHz output.
- ◆ IN\_OFF selects the internal clock source and disables the 10M output.

**QUERY SYNTAX**

:SYSTem:CLOCK?

**RESPONSE FORMAT**

<source>

<source>:= {EXT|IN\_ON|IN\_OFF}

**EXAMPLE**

The following command sets the oscilloscope clock source to inner and turns on the 10 MHz output.

Command message:

*:SYSTem:CLOCK IN\_ON*  
*SYST:CLOC IN\_ON*

Query message:

*SYST:CLOC?*

Response message:

*IN\_ON*

**:SYSTem:COMMunicate:LAN:GATeway**

**Command/Query**

**DESCRIPTION**

The command is used to set the gateway of the internal network of the oscilloscope.

The query returns the gateway of the network.

**COMMAND SYNTAX**

:SYSTem:COMMunicate:LAN:GATeway <string>

<string>:=quoted string of ASCII text.

**QUERY SYNTAX**

:SYSTem:COMMunicate:LAN:GATeway?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the gateway of the oscilloscope's internal network to "10.12.0.1".

Command message:

```
:SYSTem:COMMunicate:LAN:GATeway "10.12.0.1"  
SYST:COMM:LAN:GAT "10.12.0.1"
```

Query message:

```
SYST:COMM:LAN:GAT?
```

Response message:

```
"10.12.0.1"
```

**RELATED COMMANDS**

```
:SYSTem:COMMunicate:LAN:IPADdress  
:SYSTem:COMMunicate:LAN:SMASk  
:SYSTem:COMMunicate:LAN:TYPE
```



**:SYSTem:COMMunicate:LAN:IPADdress****Command/Query****DESCRIPTION**

The command sets the IP address of the oscilloscope's internal network interface.

The query returns the IP address of the oscilloscope's internal network interface.

**COMMAND SYNTAX**

:SYSTem:COMMunicate:LAN:IPADdress <string>

<string>:=quoted string of ASCII text.

**QUERY SYNTAX**

:SYSTem:COMMunicate:LAN:IPADdress?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the IP address of the oscilloscope's internal network interface to "10.12.255.229".

Command message:

```
:SYSTem:COMMunicate:LAN:IPADdress "10.12.255.229"  
SYST:COMM:LAN:IPAD "10.12.255.229"
```

Query message:

```
SYST:COMM:LAN:IPAD?
```

Response message:

```
"10.12.255.229"
```

**RELATED COMMANDS**

:SYSTem:COMMunicate:LAN:GATeway

:SYSTem:COMMunicate:LAN:SMASK

:SYSTem:COMMunicate:LAN:TYPE

**:SYSTem:COMMunicate:LAN:MAC****Query****DESCRIPTION**

The query returns the MAC address of the oscilloscope.

**QUERY SYNTAX**

:SYSTem:COMMunicate:LAN:MAC?

**RESPONSE FORMAT**

<byte1>:<byte2>:<byte3>:<byte4>:<byte5>:<byte6>

**EXAMPLE**

The following query returns the MAC address of the oscilloscope.

Query message:

```
SYST:COMM:LAN:MAC?
```

Response message:

```
00:01:D2:0C:00:A0
```

**:SYSTem:COMMunicate:LAN:SMASK****Command/Query****DESCRIPTION**

The command sets the subnet mask of the oscilloscope's internal network interface.

The query returns the subnet mask of the oscilloscope's internal network interface.

**COMMAND SYNTAX**

:SYSTem:COMMunicate:LAN:SMASK <string>

<string>:=quoted string of ASCII text.

**QUERY SYNTAX**

:SYSTem:COMMunicate:LAN:SMASK?

**RESPONSE FORMAT**

<string>

**EXAMPLE**

The following command sets the subnet mask of the oscilloscope's internal network interface to "10.12.255.229" .

Command message:

```
:SYSTem:COMMunicate:LAN:SMASK "255.255.0.0"  
SYST:COMM:LAN:SMAS "255.255.0.0"
```

Query message:

```
SYST:COMM:LAN:SMAS?
```

Response message:

```
"255.255.0.0"
```

**RELATED COMMANDS**

```
:SYSTem:COMMunicate:LAN:GATeway  
:SYSTem:COMMunicate:LAN:IPADdress  
:SYSTem:COMMunicate:LAN:TYPE
```

**:SYSTem:COMMunicate:LAN:TYPE****Command/Query****DESCRIPTION**

The command sets the type of LAN configuration settings.

The query returns the current type of the LAN configuration settings.

**COMMAND SYNTAX**

:SYSTem:COMMunicate:LAN:TYPE <state>

<state>:= {STATIC|DHCP}

- ♦ STATIC means that the Ethernet settings will be configured manually, using commands :SYSTem:COMMunicate:LAN:IPADdress, :SYSTem:COMMunicate:LAN:SMASK, and :SYSTem:COMMunicate:LAN:GATeway
- ♦ DHCP means that the oscilloscope's IP address, subnet mask and gateway settings will be received from a DHCP server on the local network.

**QUERY SYNTAX**

:SYSTem:COMMunicate:LAN:TYPE?

**RESPONSE FORMAT**

<state>

<state>:= {STATIC|DHCP}

**EXAMPLE**

The following command sets the type of the LAN configuration to DHCP.

Command message:

```
:SYSTem:COMMunicate:LAN:TYPE DHCP  
SYST:COMM:LAN:TYPE DHCP
```

Query message:

```
SYST:COMM:LAN:TYPE?
```

Response message:

```
DHCP
```

**RELATED COMMANDS**

```
:SYSTem:COMMunicate:LAN:GATeway  
:SYSTem:COMMunicate:LAN:IPADdress  
:SYSTem:COMMunicate:LAN:SMASK
```

**:SYSTem:COMMunicate:VNCPort****Command/Query****DESCRIPTION**

The command sets the VNC port of the oscilloscope.

The query returns the current VNC port of the oscilloscope.

**COMMAND SYNTAX**

:SYSTem:COMMunicate:VNCPort <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [5900, 5999].

**QUERY SYNTAX**

:SYSTem:COMMunicate:VNCPort?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the VNC port to 5903.

Command message:

```
:SYSTem:COMMunicate:VNCPort 5903  
SYST:COMM:VNCP 5903
```

Query message:

```
SYST:COMM:VNCP?
```

Response message:

```
5903
```

**:SYSTem:DATE****Command/Query****DESCRIPTION**

The command sets the system date of the oscilloscope.

This query returns the oscilloscope current date.

**COMMAND SYNTAX**

:SYSTem:DATE <date>

<date>:= 8-digit NR1 format, from high to low, is expressed as a 4-digit year, 2-digit month, and 2-digit day.

**QUERY SYNTAX**

:SYSTem:DATE?

**RESPONSE FORMAT**

<date>

**EXAMPLE**

The following command sets the oscilloscope current date to December 20, 2019.

Command message:

```
:SYSTem:DATE 20191220  
SYST:DATE 20190819
```

Query message:

```
SYST:DATE?
```

Response message:

```
20190819
```

**RELATED COMMANDS**

:SYSTem:TIME

**:SYSTem:EDUMode****Command/Query****DESCRIPTION**

The command sets the education mode (locks of AutoSetup, measure and cursors) of the oscilloscope.

The query returns the education mode of the oscilloscope.

**COMMAND SYNTAX**

:SYSTem:EDUMode <func>,<lock>

<func>:= {AUTOSet|MEASure|CURSor}

<lock>:= {ON|OFF}

- ◆ ON means the enable the function.
- ◆ OFF means disable the function.

**QUERY SYNTAX**

:SYSTem:EDUMode?

:SYSTem:EDUMode? <func>

**Note:**

The query without parameters will return the lock status of all functions.

**RESPONSE FORMAT**

Format 1:

AUTOSet,<lock>;MEASure,<lock>;CURSor,<lock>

Format 2:

<lock>

<lock>:= {ON|OFF}

**EXAMPLE**

The following command disables the AutoSetup function.

Command message:

*:SYSTem:EDUMode AUTOSet,OFF*

*SYST:EDUM AUTOS,OFF*

Query message:

*SYST:EDUM? AUTOS*

Response message:

*OFF*

**:SYSTem:LANGuage**

**Command/Query**

**DESCRIPTION**

The command selects the oscilloscope language display.

This query returns the oscilloscope language display.

**COMMAND SYNTAX**

:SYSTem:LANGuage <language>

<language>:=  
 {SCHinese|TCHinese|ENGLish|FRENch|JAPanese|KORean|D  
 EUTsch|ESPan|RUSSian|ITALiana|PORTuguese}

**QUERY SYNTAX**

:SYSTem:LANGuage?

**RESPONSE FORMAT**

<language>

<language>:=  
 {SCHinese|TCHinese|ENGLish|FRENch|JAPanese|KORean|D  
 EUTsch|ESPan|RUSSian|ITALiana|PORTuguese}

**EXAMPLE**

The following command sets the Oscilloscope language to English.

Command message:

*:SYSTem:LANGuage ENGLish*  
*SYST:LANG ENGL*

Query message:

*SYST:LANG?*

Response message:

*ENGLish*

**:SYSTem:MENU****Command/Query****DESCRIPTION**

The command sets the state of the menu.

The query returns the current state of the menu.

**Note:**

This command is only valid for models with the menu switch.

**COMMAND SYNTAX**

:SYSTem:MENU <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SYSTem:MENU?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the menu.

Command message:

```
:SYSTem:MENU ON  
SYST:MENU ON
```

Query message:

```
SYST:MENU?
```

Response message:

```
ON
```



**:SYSTem:NStorage****Command/Query****DESCRIPTION**

This command attempts to mount the network drive specified by the parameters.

This query returns the parameters of the mounted network drive.

**COMMAND SYNTAX**

```
:SYSTem:NStorage  
<path>,<user>,<pwd>,<anon>,<auto_con>,<rem_path>,<rem_user>,<rem_pwd>
```

<path>:= Quoted string of the server path to be mounted  
<user>:= Quoted string of the user name.  
<pwd>:= Quoted string of the user password  
<anon>:= Anonymous flag, 1 for ON while 0 for OFF  
<auto\_con>:= Automatic connection flag, 1 for ON while 0 for OFF  
<rem\_path>:= Remember path flag, 1 for ON while 0 for OFF  
<rem\_user>:= Remember user flag, 1 for ON while 0 for OFF  
<rem\_pwd>:= Remember password flag, 1 for ON while 0 for OFF

**QUERY SYNTAX**

```
:SYSTem:NStorage?
```

**RESPONSE FORMAT**

```
<path>,<user>,<pwd>,<anon>,<auto_con>,<rem_path>,<rem_user>,<rem_pwd>
```

**Note:**

For security, the password is always returned "\*\*\*\*".

**EXAMPLE**

The following command sets the network drive mount information.

Command message:

```
:SYSTem:NStorage "//10.12.255.239/nfs", "", "", 0,0,1,0,0  
SYST:NST "//10.12.255.239/nfs", "", "", 0,0,1,0,0
```

Query message:

```
SYST:NST?
```

Response message:

```
//10.12.255.239/nfs", "", "****", 0,0,1,0,0
```

**:SYSTem:NSTorage:CONNect****Command**

**DESCRIPTION** This command attempts to mount the network drive.

**COMMAND SYNTAX** :SYSTem:NSTorage:CONNect

**EXAMPLE** The following command mounts the network drive.

Command message:  
*:SYSTem:NSTorage:CONNect*  
*SYST:NST:CONN*

**:SYSTem:NSTorage:DISConnect****Command**

**DESCRIPTION** This command attempts to un-mount the network drive.

**COMMAND SYNTAX** :SYSTem:NSTorage:DISConnect

**EXAMPLE** The following command unmounts the network drive.

Command message:  
*:SYSTem:NSTorage:DISConnect*  
*SYST:NST:DISC*

**:SYSTem:NSTorage:STATus****Query**

**DESCRIPTION** The query returns the mount status of network drive.

**QUERY SYNTAX** :SYSTem:NSTorage:STATus?

**RESPONSE FORMAT** <status>  
<status>:= {ON|OFF}.

**EXAMPLE** The following query returns the mount status of network drive.

Query message:  
*SYST:NST:STAT?*

Response message:  
*OFF*

**:SYSTem:PON**

**Command/Query**

**DESCRIPTION**

The command sets the state of the Power-On-Line function. When enabled, the instrument will reboot automatically if the power is removed and re-established.

The query returns the current state of the Power-On-Line function.

**COMMAND SYNTAX**

:SYSTem:PON <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SYSTem:PON?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command sets the state of the Power-On-Line to on.

Command message:

*:SYSTem:PON ON*  
*SYST:PON ON*

Query message:

*SYST:PON?*

Response message:

*ON*

**:SYSTem:REBoot**

**Command**

**DESCRIPTION**

The command restarts the oscilloscope.

**COMMAND SYNTAX**

:SYSTem:REBoot

**EXAMPLE**

The following command restarts the oscilloscope.

Command message:

*:SYSTem:REBoot*  
*SYST:REB*

**RELATED COMMANDS**

:SYSTem:SHUTdown

**:SYSTem:REMOte****Command/Query****DESCRIPTION**

The command sets the status of the remote control. When the remote control is turned on, the touch screen, the front panel and the touch screen, front panel and peripheral will be locked, and there will be a remote prompt on the screen.

This query returns the current status of the remote setting.

**COMMAND SYNTAX**

:SYSTem:REMOte <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SYSTem:REMOte?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the remote setting.

Command message:

```
:SYSTem:REMOte ON  
SYST:REM ON
```

Query message:

```
SYST:REM?
```

Response message:

```
ON
```

**:SYSTem:SELFCal**

**Command/Query**

**DESCRIPTION**

The command instructs the oscilloscope to perform self-calibration.

The query returns the oscilloscope self-calibration status.

**COMMAND SYNTAX**

:SYSTem:SELFCal

**QUERY SYNTAX**

:SYSTem:SELFCal?

**RESPONSE FORMAT**

<state>

<state>:= {DOING|DONE}

**EXAMPLE**

The following command asks for the oscilloscope self-cal status.

Command message:

*:SYSTem:SELFCal*  
*SYST:SELFC*

Query message:

*SYST:SELFC?*

Response message:

*DONE*

**:SYSTem:SHUTdown**

**Command**

**DESCRIPTION**

The command shut down the oscilloscope.

**COMMAND SYNTAX**

:SYSTem:SHUTdown

**EXAMPLE**

The following command shut down the oscilloscope.

Command message:

*:SYSTem:SHUTdown*  
*SYST:SHUT*

**RELATED COMMANDS**

:SYSTem:REBoot

**:SYSTem:SSAVer****Command/Query****DESCRIPTION**

The command controls the automatic screensaver, which automatically shuts down the internal color monitor after a preset time.

The query returns whether the automatic screensaver feature is on.

**COMMAND SYNTAX**

:SYSTem:SSAVer <time>

<time>:= {OFF|1MIN|5MIN|10MIN|30MIN|60MIN}

**QUERY SYNTAX**

:SYSTem:SSAVer?

**RESPONSE FORMAT**

<time>

<time>:= {OFF|1MIN|5MIN|10MIN|30MIN|60MIN}

**EXAMPLE**

The following command sets the automatic screensaver to 10 minutes.

Command message:

*:SYSTem:SSAVer 10MIN*

*SYST:SSAV 10MIN*

Query message:

*SYST:SSAV?*

Response message:

*10MIN*

**:SYSTem:TIME****Command/Query****DESCRIPTION**

The command sets the oscilloscope current time using a 24-hour format.

This query returns the oscilloscope current time.

**COMMAND SYNTAX**

:SYSTem:TIME <time>

<time>:= 8-digit NR1 format, from high to low, is expressed as 2-digit hour, 2-digit minute, and 2-digit second.

**QUERY SYNTAX**

:SYSTem:TIME?

**RESPONSE FORMAT**

<time>

**EXAMPLE**

The following command sets the current time of the oscilloscope to 08:10:40.

Command message:

```
:SYSTem:TIME 081040  
SYST:TIME 081040
```

Query message:

```
SYST:TIME?
```

Response message:

```
081040
```

**RELATED COMMANDS**

:SYSTem:DATE

**:SYSTem:TOUCh****Command/Query****DESCRIPTION**

The command sets the status of the touch screen.

The query returns the current status of the touch screen.

**COMMAND SYNTAX**

:SYSTem:TOUCh <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:SYSTem:TOUCh?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command enables the touch setting.

Command message:

*:SYSTem:TOUCh ON*  
*SYST:TOUC ON*

Query message:

*SYST:TOUC?*

Response message:

*ON*



## **TIMEbase Commands**

The :TIMEBASE subsystem commands control the horizontal (X-axis) functions. The time per division, delay, and reference can be controlled for the main and window (zoomed) time bases.

- ◆ **:TIMEbase:DElay**
- ◆ **:TIMEbase:REFerence**
- ◆ **:TIMEbase:REFerence:POSition**
- ◆ **:TIMEbase:SCALe**
- ◆ **:TIMEbase:WINDow**
- ◆ **:TIMEbase:WINDow:DElay**
- ◆ **:TIMEbase:WINDow:SCALe**

**:TIMebase:DELay****Command/Query****DESCRIPTION**

The command specifies the main timebase delay. This delay is the time between the trigger event and the delay reference point on the screen.

The query returns the current delay value.

**COMMAND SYNTAX**

:TIMebase:DELay <delay\_value>

<delay\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [-5000div\*timebase, 5div\*timebase].

**QUERY SYNTAX**

:TIMebase:DELay?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command specifies a 10 us delay of main time base.

Command message:

*:TIMebase:DELay 1.00E-05*

*TIM:DEL 1.00E-05*

Query message:

*TIM:DEL?*

Response message:

*1.00E-05*

**RELATED COMMANDS**

:TIMebase:SCALe

**:TIMebase:REFerence****Command/Query****DESCRIPTION**

The command sets the strategy for the delay value change in the horizontal direction when the horizontal scale is changed.

The query returns the current horizontal reference strategy.

**COMMAND SYNTAX**

:TIMebase:REFerence <type>

<type>:= {DELay|POSition}

- ◆ DELay means when the time base is changed, the horizontal delay value remains fixed. As the horizontal timebase scale is changed, the waveform expands/contracts around the center of the display.
- ◆ POSition means When the time base is changed, the horizontal delay remains fixed to the grid position on the display. As the horizontal time base scale is changed, the waveform expands/contracts around the position of the horizontal display.

**QUERY SYNTAX**

:TIMebase:REFerence?

**RESPONSE FORMAT**

<type>

<type>:= {DELay|POSition}

**EXAMPLE**

The following command sets the type of the horizontal reference to DELay.

Command message:

*:TIMebase:REFerence DELay*  
*TIM:REF DEL*

Query message:

*TIM:REF?*

Response message:

*DELay*

**:TIMebase:REFerence:POSition****Command/Query****DESCRIPTION**

The command sets the horizontal reference center when the reference strategy is DELay.

The query returns the current horizontal reference center.

**COMMAND SYNTAX**

:TIMebase:REFerence:POSition <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 100].

**QUERY SYNTAX**

:TIMebase:REFerence:POSition?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command changes the horizontal reference center to 20%.

Command message:

*:TIMebase:REFerence:POSition 20*

*TIM:REF:POS 20*

Query message:

*TIM:REF:POS?*

Response message:

*20*

**RELATED COMMANDS**

:TIMebase:REFerence

**:TIMebase:SCALe****Command/Query****DESCRIPTION**

The command sets the horizontal scale per division for the main window.

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

**Note:**

Due to the limitation of the expansion strategy, when the time base is set from large to small, it will automatically adjust to the minimum time base that can be set currently.

**COMMAND SYNTAX**

:TIMebase:SCALe <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The range of value varies from the models. See the datasheet for details.

**QUERY SYNTAX**

:TIMebase:SCALe?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the horizontal scale to 100 ns/div.

Command message:

```
:TIMebase:SCALe 1.00E-07
```

```
TIM:SCAL 1.00E-07
```

Query message:

```
TIM:SCAL?
```

Response message:

```
1.00E-07
```

**RELATED COMMANDS**

:TIMebase:DELay

**:TIMebase:WINDow****Command/Query****DESCRIPTION**

The command turns on or off the zoomed window.

The query returns the state of the zoomed window.

**COMMAND SYNTAX**

:TIMebase:WINDow <state>

<state>:= {ON|OFF}

**QUERY SYNTAX**

:TIMebase:WINDow?

**RESPONSE FORMAT**

<state>

<state>:= {ON|OFF}

**EXAMPLE**

The following command turns on the zoomed window.

Command message:

```
:TIMebase:WINDow ON  
TIM:WIND ON
```

Query message:

```
TIM:WIND?
```

Response message:

```
ON
```

**RELATED COMMANDS**

```
:TIMebase:WINDow:DELay  
:TIMebase:WINDow:SCALE
```

**:TIMebase:WINDow:DELay****Command/Query****DESCRIPTION**

The command sets the horizontal position in the zoomed view of the main sweep.

The query returns the current delay value between the zoomed window and the main sweep.

**COMMAND SYNTAX**

:TIMebase:WINDow:DELay <delay\_value>

<delay\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

- The main sweep range and the main sweep horizontal position determine the range for the delay value of the zoomed window. It must keep the zoomed view window within the main sweep range.
- If you set the delay to a value outside of the legal range, the delay value is automatically set to the nearest legal value.

**QUERY SYNTAX**

:TIMebase:WINDow:DELay?

**RESPONSE FORMAT**

<delay\_value>

<delay\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets 1 ms delay value to change the position of the zoomed window.

Command message:

```
:TIMebase:WINDow:DELay 1.00E-03  
TIM:WIND:DEL 1.00E-03
```

Query message:

```
TIM:WIND:DEL?
```

Response message:

```
1.00E-03
```

**RELATED COMMANDS**

```
:TIMebase:WINDow:SCALE  
:TIMebase:SCALE  
:TIMebase:DELay
```

**:TIMebase:WINDow:SCALe****Command/Query****DESCRIPTION**

The command sets the zoomed window horizontal scale (seconds/division).

The query returns the current zoomed window scale setting.

**COMMAND SYNTAX**

:TIMebase:WINDow:SCALe <scale\_value>

<scale\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**Note:**

The scale of the zoomed window cannot be greater than that of the main window. If you set the value greater than, it will automatically be set to the same value as the main window.

**QUERY SYNTAX**

:TIMebase:WINDow:SCALe?

**RESPONSE FORMAT**

<scale\_value>

<scale\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets a 1 ms/div horizontal scale for the zoomed window.

Command message:

```
:TIMebase:WINDow:SCALe 1.00E-03  
TIM:WIND:SCAL 1.00E-03
```

Query message:

```
TIM:WIND:SCAL?
```

Response message:

```
1.00E-03
```

**RELATED COMMANDS**

```
:TIMebase:WINDow:DELay  
:TIMebase:SCALe  
:TIMebase:DELay
```



## TRIGger Commands

The :TRIGGER subsystem commands control the trigger modes and parameters for each trigger type.

- ◆ :TRIGger:FREQuency
- ◆ :TRIGger:MODE
- ◆ :TRIGger:RUN
- ◆ :TRIGger:STATus
- ◆ :TRIGger:STOP
- ◆ :TRIGger:TYPE
- ◆ :TRIGger:EDGE Commands
- ◆ :TRIGger:SLOPe Commands
- ◆ :TRIGger:PULSe Commands
- ◆ :TRIGger:VIDeo Commands
- ◆ :TRIGger:WINDow Commands
- ◆ :TRIGger:INTerval Commands
- ◆ :TRIGger:DROPOut Commands
- ◆ :TRIGger:PATTern Commands
- ◆ :TRIGger:QUALified Commands
- ◆ :TRIGger:DELAy Commands
- ◆ :TRIGger:NEDGE Commands
- ◆ :TRIGger:SHOLd Commands
- ◆ :TRIGger:IIC Commands
- ◆ :TRIGger:SPI Commands
- ◆ :TRIGger:UART Commands
- ◆ :TRIGger:CAN Commands
- ◆ :TRIGger:LIN Commands
- ◆ :TRIGger:FLEXray Commands [Option]
- ◆ :TRIGger:CANFd Commands [Option]
- ◆ :TRIGger:IIS Commands [Option]

**:TRIGger:FREQuency****Query****DESCRIPTION**

The query returns the value of hardware frequency counter in hertz if available. The default precision of the returned frequency is 3 digits, and the maximum valid precision is 7 digits. Use the command “:FORMat:DATA” to set the data precision.

**QUERY SYNTAX**

:TRIGger:FREQuency?

**RESPONSE FORMAT**

<val>

<val>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command returns the value of hardware frequency counter.

Command message:

*:FORMat:DATA CUSTom,7  
TRIG:FREQ?*

Response message:

*1.234561E+04*

**RELATED COMMANDS**

:FORMat:DATA

**:TRIGger:MODE****Command/Query****DESCRIPTION**

The command sets the mode of the trigger.

The query returns the current mode of trigger.

**COMMAND SYNTAX**

:TRIGger:MODE <mode>

<mode>:= {SINGle|NORMal|AUTO|FTRIG}

- ◆ **AUTO**: The oscilloscope begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state on the top left corner of the user interface shows Trig'd, and the interface shows stable waveform. Otherwise, the running state always shows Auto, and the interface shows unstable waveform.
- ◆ **NORMal**: The oscilloscope enters the wait trigger state and begins to search for trigger signals that meet the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Otherwise, the running state shows Ready, and the interface displays the last triggered waveform (previous trigger) or does not display the waveform (no previous trigger).
- ◆ **SINGle**: The backlight of SINGLE key lights up, the oscilloscope enters the waiting trigger state and begins to search for the trigger signal that meets the conditions. If the trigger signal is satisfied, the running state shows Trig'd, and the interface shows stable waveform. Then, the oscilloscope stops scanning, the RUN/STOP key becomes red, and the running status shows Stop. Otherwise, the running state shows Ready, and the interface does not display the waveform.
- ◆ **FTRIG**: Force to acquire a frame regardless of whether the input signal meets the trigger conditions or not.

**QUERY SYNTAX**

:TRIGger:MODE?

**RESPONSE FORMAT**

<mode>

<mode>:= {SINGle|NORMal|AUTO|FTRIG}

**EXAMPLE**

The following command sets the oscilloscope to SINGLE trigger mode.

Command message:

```
:TRIGger:MODE SINGle  
TRIG:MODE SING
```

Query message:

```
TRIG:MODE?
```

Response message:

```
SINGle
```

**:TRIGger:RUN****Command****DESCRIPTION**

The command sets the oscilloscope to run.

**COMMAND SYNTAX**

:TRIGger:RUN

**EXAMPLE**

The following command sets the oscilloscope to run.

Command message:

*:TRIGger:RUN*

*TRIG:RUN*

**RELATED COMMANDS**

:TRIGger:STOP

**:TRIGger:STATus****Query****DESCRIPTION**

The command query returns the current state of the trigger.

**QUERY SYNTAX**

:TRIGger:STATus?

**RESPONSE FORMAT**

<status>

<status>:= {Arm|Ready|Auto|Trig'd|Stop|Roll}

**EXAMPLE**

The following command queries the state of trigger mode.

Query message:

*TRIG:STAT?*

Response message:

*Stop*

**RELATED COMMANDS**

:TRIGger:MODE

**:TRIGger:STOP****Command****DESCRIPTION**

The command sets the oscilloscope from run to stop.

**COMMAND SYNTAX**

:TRIGger:STOP

**EXAMPLE**

The following command stops the oscilloscope.

Command message:

*:TRIGger:STOP*

*TRIG:STOP*

**RELATED COMMANDS**

:TRIGger:RUN

**:TRIGger:TYPE****Command/Query****DESCRIPTION**

The command sets the type of trigger.

The query returns the current type of trigger.

**COMMAND SYNTAX**

:TRIGger:TYPE <type>

<type>:= {EDGE|PULSE|SLOPe|INTErval|PATTErn|RUNT|WINDow|DROPOut|VIDeo|QUALified|NEDGE|DELay|SHOLd|IIC|SPI|UART|LIN|CAN|FLEXray|CANFd|IIS|M1553|SENT|A429}

**QUERY SYNTAX**

:TRIGger:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {EDGE|PULSE|SLOPe|INTErval|PATTErn|RUNT|WINDow|DROPOut|VIDeo|QUALified|NEDGE|DELay|SHOLd|IIC|SPI|UART|LIN|CAN|FLEXray|CANFd|IIS|M1553|SENT||A429}

**EXAMPLE**

The following command sets the type of trigger to edge trigger.

Command message:

*:TRIGger:TYPE EDGE*

*TRIG:TYPE EDGE*

Query message:

*TRIG:TYPE?*

Command message:

*EDGE*

**:TRIGger:EDGE Commands**

The :TRIGGER:EDGE subsystem commands control the edge trigger parameters.

- ◆ **:TRIGger:EDGE:COUPling**
- ◆ **:TRIGger:EDGE:HLDEvent**
- ◆ **:TRIGger:EDGE:HLDTIME**
- ◆ **:TRIGger:EDGE:HOLDoff**
- ◆ **:TRIGger:EDGE:HStart**
- ◆ **:TRIGger:EDGE:IMPedance**
- ◆ **:TRIGger:EDGE:LEVel**
- ◆ **:TRIGger:EDGE:NREJect**
- ◆ **:TRIGger:EDGE:SLOPe**
- ◆ **:TRIGger:EDGE:SOURce**

**:TRIGger:EDGE:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the edge trigger.

The query returns the current coupling mode of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter that adds a low-pass filter in the trigger path to remove high-frequency components from the trigger waveform. Use the high-frequency rejection filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:EDGE:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets the coupling mode of the edge trigger to DC.

Command message:

*:TRIGger:EDGE:COUPling DC*  
*TRIG:EDGE:COUP DC*

Query message:

*TRIG:EDGE:COUP?*

Response message:

*DC*

**:TRIGger:EDGE:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the edge trigger.

The query returns the current number of holdoff events of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:EDGE:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the edge trigger to 3.

Command message:

```
:TRIGger:EDGE:HLDEvent 3  
TRIG:EDGE:HLDEV 3
```

Query message:

```
TRIG:EDGE:HLDEV?
```

Response message:

```
3
```

**RELATED COMMANDS**

:TRIGger:EDGE:HOLDoff



**:TRIGger:EDGE:HLDTIME**

**Command/Query**

**DESCRIPTION**

The command sets the holdoff time of the edge trigger.

The query returns the current holdoff time of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A<br>SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [8.00E-09, 3.00E+01] |
| SHS800X/SHS1000X  | [80.00E-09, 1.5E+00] |

**QUERY SYNTAX**

:TRIGger:EDGE:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the holdoff time of the edge trigger to 15 ns.

Command message:

*:TRIGger:EDGE:HLDTIME 1.50E-08*  
*TRIG:EDGE:HLDT 1.50E-08*

Query message:

*TRIG:EDGE:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:DROPOut:HOLDoff

**:TRIGger:EDGE:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the edge trigger.

The query returns the current holdoff type of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the number of trigger events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:EDGE:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the edge trigger.

Command message:

```
:TRIGger:EDGE:HOLDoff OFF  
TRIG:EDGE:HOLD OFF
```

Query message:

```
TRIG:EDGE:HOLD?
```

Response message:

```
OFF
```

**RELATED COMMANDS**

:TRIGger:EDGE:HLDEvent

:TRIGger:EDGE:HLDTime

:TRIGger:EDGE:HSTart

**:TRIGger:EDGE:HStart****Command/Query****DESCRIPTION**

The command defines the initial position of the edge trigger holdoff.

The query returns the initial position of the edge trigger holdoff.

**COMMAND SYNTAX**

:TRIGger:EDGE:HStart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:EDGE:HStart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to last trigger.

Command message:

*:TRIGger:EDGE:HStart LAST\_TRIG  
TRIG:EDGE:HST LAST\_TRIG*

Query message:

*TRIG:EDGE:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:EDGE:HOLDoff

**:TRIGger:EDGE:IMPedance****Command/Query****DESCRIPTION**

The command sets the edge trigger source impedance, which is only valid when the source is EXT or EXT/5.

The query returns the impedance of external trigger source.

**COMMAND SYNTAX**

:TRIGger:EDGE:IMPedance <ohm>

<ohm>:= {ONEMeg|FIFTy}

**QUERY SYNTAX**

:TRIGger:EDGE:IMPedance?

**RESPONSE FORMAT**

<ohm >

<ohm>:= {ONEMeg|FIFTy}

**EXAMPLE**

The following command sets the impedance of ext trigger source to 50 ohm.

Command message:

```
:TRIGger:EDGE:IMPedance FIFTy  
TRIG:EDGE:IMP FIFT
```

Query message:

```
TRIG:EDGE:IMP?
```

Response message:

```
FIFTy
```

**RELATED COMMANDS**

:TRIGger:EDGE:SOURce

**:TRIGger:EDGE:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the edge trigger.

The query returns the current trigger level value of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:LEVel <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:EDGE:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the trigger level of the edge trigger to 0.5 V.

Command message:

*:TRIGger:EDGE:LEVel 5.00E-01*  
*TRIG:EDGE:LEV 5.00E-01*

Query message:

*TRIG:EDGE:LEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:EDGE:SOURce

**:TRIGger:EDGE:NREJect****Command/Query****DESCRIPTION**

The command sets the state of the noise rejection.

The query returns the current state of the noise rejection.

**COMMAND SYNTAX**

:TRIGger:EDGE:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:EDGE:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on noise rejection.

Command message:

*:TRIGger:EDGE:NREJect ON*  
*TRIG:EDGE:NREJ ON*

Query message:

*TRIG:EDGE:NREJ?*

Response message:

*ON*

**:TRIGger:EDGE:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the edge trigger.

The query returns the current slope setting of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing|ALternate}

**QUERY SYNTAX**

:TRIGger:EDGE:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing|ALternate}

**EXAMPLE**

The following command set the rising slope as trigger edge.

Command message:

```
:TRIGger:EDGE:SLOPe RISing  
TRIG:EDGE:SLOP RIS
```

Query message:

```
TRIG:EDGE:SLOP?
```

Response message:

```
RISing
```

**:TRIGger:EDGE:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the edge trigger.

The query returns the current trigger source of the edge trigger.

**COMMAND SYNTAX**

:TRIGger:EDGE:SOURce <source>

<source>:= {C<n>|D<d>|EX|EX5|LINE}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:EDGE:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>|EX|EX5|LINE}

**EXAMPLE**

The following command sets the trigger source of the edge trigger as C1.

Command message:

```
:TRIGger:EDGE:SOURce C1
TRIG:EDGE:SOUR C1
```

Query message:

```
TRIG:EDGE:SOUR?
```

Response message:

```
C1
```

**RELATED COMMANDS**

:TRIGger:EDGE:LEVel



## **:TRIGger:SLOPe Commands**

The :TRIGGER:SLOPe subsystem commands control the slope trigger parameters.

- ◆ **:TRIGger:SLOPe:COUPling**
- ◆ **:TRIGger:SLOPe:HLDEvent**
- ◆ **:TRIGger:SLOPe:HLDTIME**
- ◆ **:TRIGger:SLOPe:HLEVel**
- ◆ **:TRIGger:SLOPe:HOLDoff**
- ◆ **:TRIGger:SLOPe:HStArt**
- ◆ **:TRIGger:SLOPe:LIMit**
- ◆ **:TRIGger:SLOPe:LLEVel**
- ◆ **:TRIGger:SLOPe:NREJect**
- ◆ **:TRIGger:SLOPe:SLOPe**
- ◆ **:TRIGger:SLOPe:SOURce**
- ◆ **:TRIGger:SLOPe:TLOWer**
- ◆ **:TRIGger:SLOPe:TUPPer**

**:TRIGger:SLOPe:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the slope trigger.

The query returns the current the coupling mode of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a 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.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:SLOPe:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets the coupling mode of the slope trigger to DC.

Command message:

```
:TRIGger:SLOPe:COUPling DC
TRIG:SLOP:COUP DC
```

Query message:

```
TRIG:SLOP:COUP?
```

Response message:

```
DC
```

**:TRIGger:SLOPe:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the slope trigger.

The query returns the current number of holdoff events of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:SLOPe:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the slope trigger to 3.

Command message:

```
:TRIGger:SLOPe:HLDEvent 3  
TRIG:SLOP:HLDEV 3
```

Query message:

```
TRIG:SLOP:HLDEV?
```

Response message:

```
3
```

**RELATED COMMANDS**

:TRIGger:SLOPe:HOLDoff

**:TRIGger:SLOPe:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the slope trigger.

The query returns the current holdoff time of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model         | Value Range          |
|---------------|----------------------|
| SDS7000A      | [8.00E-09, 3.00E+01] |
| SDS5000X      |                      |
| SDS2000X Plus |                      |
| SDS6000 Pro   |                      |
| SDS6000A      |                      |
| SDS6000L      |                      |
| SDS3000X HD   |                      |
| SDS2000X HD   |                      |
| SDS1000X HD   |                      |
| SDS800X HD    |                      |

**QUERY SYNTAX**

:TRIGger:SLOPe:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the holdoff time of the slope trigger to 15 ns.

Command message:

*:TRIGger:SLOPe:HLDTIME 1.50E-08*  
*TRIG:SLOP:HLDT 1.50E-08*

Query message:

*TRIG:SLOP:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:SLOPe:HOLDoff

**:TRIGger:SLOPe:HLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the high level of the slope trigger.

The query returns the current high level of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:HLEVel <high\_level\_value>

<high\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset,4.26*vertical_scale-vertical_offse] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset] |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset] |

**Note:**

The high level value cannot be less than the low level value using by the command :TRIGger:SLOPe:LLEVel.

**QUERY SYNTAX**

:TRIGger:SLOPe:HLEVel?

**RESPONSE FORMAT**

<high\_level\_value>

<high\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the high level of the slope trigger to 0.5 V.

Command message:

*:TRIGger:SLOPe:HLEVel 5.00E-01*  
*TRIG:SLOP:HLEV 5.00E-01*

Query message:

*TRIG:SLOP:HLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:SLOPe:LLEVel

**:TRIGger:SLOPe:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the slope trigger.

The query returns the current holdoff type of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry

**QUERY SYNTAX**

:TRIGger:SLOPe:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type>:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the slope trigger.

Command message:

```
:TRIGger:SLOPe:HOLDoff OFF
TRIG:SLOP:HOLD OFF
```

Query message:

```
TRIG:SLOP:HOLD?
```

Response message:

```
OFF
```

**RELATED COMMANDS**

```
:TRIGger:SLOPe:HLTime
:TRIGger:SLOPe:HLDEvent
:TRIGger:SLOPe:HStart
```

**:TRIGger:SLOPe:HSTart****Command/Query****DESCRIPTION**

The command defines the initial position of the slope trigger holdoff.

The query returns the initial position of the slope trigger holdoff.

**COMMAND SYNTAX**

:TRIGger:SLOPe:HSTart <type>

<start\_type>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:SLOPe:HSTart?

**RESPONSE FORMAT**

<type>

<type>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to LAST\_TRIG (last trigger).

Command message:

*:TRIGger:SLOPe:HSTart LAST\_TRIG  
TRIG:SLOP:HST LAST\_TRIG*

Query message:

*TRIG:SLOP:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:SLOPe:HOLDoff

**:TRIGger:SLOPe:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the slope trigger.

The query returns the current limit range type of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:LIMit <type>

<type>:= {LESSthan|GREATerthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:SLOPe:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATerthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the slope trigger to LESSthan.

Command message:

```
:TRIGger:SLOPe:LIMit LESSthan  
TRIG:SLOP:LIM LESS
```

Query message:

```
TRIG:SLOP:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:TRIGger:SLOPe:TLOWer

:TRIGger:SLOPe:TUPPer



**:TRIGger:SLOPe:LLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the low level of the slope trigger.

The query returns the current low level of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:LLEVel <low\_level\_value>

<low\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset,4.26*vertical_scale-vertical_offse] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset] |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset] |

**Note:**

The low level value cannot be greater than the low level value using by the command :TRIGger:SLOPe:HLEVel.

**QUERY SYNTAX**

:TRIGger:SLOPe:LLEVel?

**RESPONSE FORMAT**

<low\_level\_value>

<low\_level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the low level of the slope trigger to -0.5 V.

Command message:

*:TRIGger:SLOPe:LLEVel -5.00E-01  
TRIG:SLOP:LLEV -5.00E-01*

Query message:

*TRIG:SLOP:LLEV?*

Response message:

*-5.00E-01*

**RELATED COMMANDS**

:TRIGger:SLOPe:HLEVel

**:TRIGger:SLOPe:NREJect****Command/Query****DESCRIPTION**

The command sets the state of noise rejection.

The query returns the current state of noise rejection.

**COMMAND SYNTAX**

:TRIGger:SLOPe:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:SLOPe:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the noise rejection.

Command message:

*:TRIGger:SLOPe:NREJect ON*

*TRIG:SLOP:NREJ ON*

Query message:

*TRIG:SLOP:NREJ?*

Response message:

*ON*

**:TRIGger:SLOPe:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the slope trigger.

The query returns the current slope of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:SLOPe:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the slope trigger.

Command message:

*:TRIGger:SLOPe:SLOPe RISing*  
*TRIG:SLOP:SLOP RIS*

Query message:

*TRIG:SLOP:SLOP?*

Response message:

*RISing*

**:TRIGger:SLOPe:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the slope trigger.

The query returns the current trigger source of the slope trigger.

**COMMAND SYNTAX**

:TRIGger:SLOPe:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SLOPe:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the trigger source of the slope trigger to C2 (channel 2).

Command message:

*:TRIGger:SLOPe:SOURce C2*

*TRIG:SLOP:SOUR C2*

Query message:

*TRIG:SLOP:SOUR?*

Response message:

*C2*

**:TRIGger:SLOPe:TLOWer**  
**Command/Query**

**DESCRIPTION**

The command sets the lower value of the slope trigger limit type.  
 The query returns the current lower value of the slope trigger limit type.

**COMMAND SYNTAX**

:TRIGger:SLOPe:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:SLOPe:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:SLOPe:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the slope trigger to 10 ns.

Command message:

*:TRIGger:SLOPe:TLOWer 1.00E-08*  
*TRIG:SLOP:TLOW 1.00E-08*

Query message:

*TRIG:SLOP:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:SLOPe:LIMit  
 :TRIGger:SLOPe:TUPPer

**:TRIGger:SLOPe:TUPPer**  
**Command/Query**

**DESCRIPTION**

The command sets the upper value of the slope trigger limit type.

The query returns the current upper value of the slope trigger limit type.

**COMMAND SYNTAX**

:TRIGger:SLOPe:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:SLOPe:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:SLOPe:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper value of the slope trigger to 30 ns, when the limit range type is OUTer.

Command message:

*:TRIGger:SLOPe:TUPPer 3.00E-08*  
*TRIG:SLOP:TUPP 3.00E-08*

Query message:

*TRIG:SLOP:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:TRIGger:SLOPe:LIMit  
:TRIGger:SLOPe:TLOWer

## **:TRIGger:PULSe Commands**

The :TRIGGER:PULSE subsystem commands control the pulse trigger parameters.

- ◆ **:TRIGger:PULSe:COUPLing**
- ◆ **:TRIGger:PULSe:HLDEvent**
- ◆ **:TRIGger:PULSe:HLDTime**
- ◆ **:TRIGger:PULSe:HOLDoff**
- ◆ **:TRIGger:PULSe:HStart**
- ◆ **:TRIGger:PULSe:LEVel**
- ◆ **:TRIGger:PULSe:LIMit**
- ◆ **:TRIGger:PULSe:NREJect**
- ◆ **:TRIGger:PULSe:POLarity**
- ◆ **:TRIGger:PULSe:SOURce**
- ◆ **:TRIGger:PULSe:TLOWer**
- ◆ **:TRIGger:PULSe:TUPPer**

**:TRIGger:PULSe:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the pulse trigger.

The query returns the coupling mode of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a low-pass filter in the trigger path to remove high frequency components from the trigger waveform. Use the high-frequency rejection filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:PULSe:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets coupling mode of the pulse trigger to DC.

Command message:

```
:TRIGger:PULSe:COUPling DC
TRIG:PULS:COUP DC
```

Query message:

```
TRIG:PULS:COUP?
```

Response message:

```
DC
```



**:TRIGger:PULSe:HLDEvent**

**Command/Query**

**DESCRIPTION**

This command sets the number of holdoff events of the pulse trigger.

The query returns the current number of holdoff events of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:PULSe:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the pulse trigger to 3.

Command message:

*:TRIGger:PULSe:HLDEvent 3*  
*TRIG:PULS:HLDEV 3*

Query message:

*TRIG:PULS:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

:TRIGger:PULSe:HOLDoff

**:TRIGger:PULSe:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the pulse trigger.

The query returns the current holdoff time of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model         | Value Range          |
|---------------|----------------------|
| SDS7000A      | [8.00E-09, 3.00E+01] |
| SDS5000X      |                      |
| SDS2000X Plus |                      |
| SDS6000 Pro   |                      |
| SDS6000A      |                      |
| SDS6000L      |                      |
| SDS3000X HD   |                      |
| SDS2000X HD   |                      |
| SDS1000X HD   |                      |
| SDS800X HD    |                      |

**QUERY SYNTAX**

:TRIGger:PULSe:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the holdoff time of the pulse trigger to 15 ns.

Command message:

*:TRIGger:PULSe:HLDTIME 1.50E-08*  
*TRIG:PULS:HLDT 1.50E-08*

Query message:

*TRIG:PULS:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:PULSe:HOLDoff

**:TRIGger:PULSe:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the pulse trigger.

The query returns the current holdoff type of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:PULSe:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type >:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the pulse trigger.

Command message:

*:TRIGger:PULSe:HOLDoff OFF*  
*TRIG:PULS:HOLD OFF*

Query message:

*TRIG:PULS:HOLD?*

Response message:

*OFF*

**RELATED COMMANDS**

:TRIGger:PULSe:HLDEvent

:TRIGger:PULSe:HLTime

:TRIGger:PULSe:HStart

**:TRIGger:PULSe:HSTart****Command/Query****DESCRIPTION**

The command defines the initial position of the pulse trigger holdoff.

The query returns the initial position of the pulse trigger holdoff.

**COMMAND SYNTAX**

:TRIGger:PULSe:HSTart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:PULSe:HSTart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode of pulse trigger to LAST\_TRIG (last trigger).

Command message:

```
:TRIGger:PULSe:HSTart LAST_TRIG
TRIG:PULS:HST LAST_TRIG
```

Query message:

```
TRIG:PULS:HST?
```

Response message:

```
LAST_TRIG
```

**RELATED COMMANDS**

:TRIGger:PULSe:HOLDoff

**:TRIGger:PULSe:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the pulse trigger.

The query returns the current trigger level of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:LEVel <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:PULSe:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the trigger level of the pulse trigger to 0.5 V.

Command message:

*:TRIGger:PULSe:LEVel 5.00E-01*  
*TRIG:PULS:LEV 5.00E-01*

Query message:

*TRIG:PULS:LEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:PULSe:SOURce

**:TRIGger:PULSe:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the pulse trigger.

The query returns the current limit range type of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:PULSe:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the trigger limit of the pulse trigger to inner.

Command message:

```
:TRIGger:PULSe:LIMit INNER  
TRIG:PULS:LIM INN
```

Query message:

```
TRIG:PULS:LIM?
```

Response message:

```
INNER
```

**RELATED COMMANDS**

:TRIGger:PULSe:TLOWer

:TRIGger:PULSe:TUPPer

**:TRIGger:PULSe:NREJect****Command/Query****DESCRIPTION**

The command sets the state of noise rejection.

The query returns the current state of the noise rejection function.

**COMMAND SYNTAX**

:TRIGger:PULSe:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:PULSe:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on noise rejection.

Command message:

*:TRIGger:PULSe:NREJect ON*

*TRIG:PULS:NREJ ON*

Query message:

*TRIG:PULS:NREJ?*

Response message:

*ON*

**:TRIGger:PULSe:POLarity****Command/Query****DESCRIPTION**

The command sets the polarity of the pulse trigger.

The query returns the current polarity of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:POLarity <polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**QUERY SYNTAX**

:TRIGger:PULSe:POLarity?

**RESPONSE FORMAT**

<polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**EXAMPLE**

The following command sets the polarity of the pulse trigger to POSitive.

Command message:

```
:TRIGger:PULSe:POLarity POSitive  
TRIG:PULS:POL POS
```

Query message:

```
TRIG:PULS:POL?
```

Response message:

```
POSitive
```



**:TRIGger:PULSe:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the pulse trigger.

The query returns the current trigger source of the pulse trigger.

**COMMAND SYNTAX**

:TRIGger:PULSe:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:PULSe:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the polarity of the pulse trigger as channel 2.

Command message:

*:TRIGger:PULSe:SOURce C2*  
*TRIG:PULS:SOUR C2*

Query message:

*TRIG:PULS:SOUR?*

Response message:

*C2*

**:TRIGger:PULSe:TLOWer**

**Command/Query**

**DESCRIPTION**

The command sets the lower value of the pulse trigger limit type.

The query returns the current lower value of the pulse trigger limit type.

**COMMAND SYNTAX**

:TRIGger:PULSe:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:PULSe:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:PULSe:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the pulse trigger to 10 ns.

Command message:

*:TRIGger:PULSe:TLOWer 1.00E-08*

*TRIG:PULS:TLOW 1.00E-08*

Query message:

*TRIG:PULS:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:PULSe:LIMit

:TRIGger:PULSe:TUPPer

**:TRIGger:PULSe:TUPPer**  
**Command/Query**

**DESCRIPTION**

The command sets the upper value of the pulse trigger limit type.

The query returns the current upper value of the pulse trigger limit type.

**COMMAND SYNTAX**

:TRIGger:PULSe:TUPPer <value>

<value>:= Value in NR3 format.The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:PULSe:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:PULSe:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the upper time of the pulse trigger to 30 ns.

Command message:

*:TRIGger:PULSe:TUPPer 3.00E-08*  
*TRIG:PULS:TUPP 3.00E-08*

Query message:

*TRIG:PULS:TUPP?*

Response message:

*3.00E-08*

**RELATED COMMANDS**

:TRIGger:PULSe:LIMit  
 :TRIGger:PULSe:TLOWer

**:TRIGger:VIDeo Commands**

The :TRIGGER:VIDeo subsystem commands control the video trigger parameters.

- ◆ **:TRIGger:VIDeo:FCNT**
- ◆ **:TRIGger:VIDeo:FIELD**
- ◆ **:TRIGger:VIDeo:FRATE**
- ◆ **:TRIGger:VIDeo:INTerlace**
- ◆ **:TRIGger:VIDeo:LCNT**
- ◆ **:TRIGger:VIDeo:LEVel**
- ◆ **:TRIGger:VIDeo:LINE**
- ◆ **:TRIGger:VIDeo:SOURce**
- ◆ **:TRIGger:VIDeo:STANdard**
- ◆ **:TRIGger:VIDeo:SYNC**

**:TRIGger:VIDeo:FCNT****Command/Query****DESCRIPTION**

The command sets the fields of the custom video trigger.

The query returns the current fields of the custom video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:FCNT <field\_cnt>

<field\_cnt>:= {1|2|4|8}

**QUERY SYNTAX**

:TRIGger:VIDeo:FCNT?

**RESPONSE FORMAT**

<field\_cnt>

<field\_cnt>:= {1|2|4|8}

**EXAMPLE**

The following command sets the fields of the custom video trigger to 8.

Command message:

*:TRIGger:VIDeo:FCNT 8*  
*TRIG:VID:FCNT 8*

Query message:

*TRIG:VID:FCNT?*

Response message:

*8*

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard

**:TRIGger:VIDeo:FIEld****Command/Query****DESCRIPTION**

The command sets the synchronous trigger field when the video standard is NTSC, PAL, 1080i/50 or 1080i/60.

The query returns the current synchronous trigger field when the video standard is NTSC, PAL, 1080i/50 or 1080i/60.

**COMMAND SYNTAX**

:TRIGger:VIDeo:FIEld <field>

<field>:= {1|2}

**QUERY SYNTAX**

:TRIGger:VIDeo:FIEld?

**RESPONSE FORMAT**

<field>

<field>:= {1|2}

**EXAMPLE**

The following command sets the synchronous trigger field to field 2 when the video standard is NTSC.

Command message:

*:TRIGger:VIDeo:FIEld 2*

*TRIG:VID:FIEL 2*

Query message:

*TRIG:VID:FIEL?*

Response message:

*2*

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard

:TRIGger:VIDeo:SYNC

**:TRIGger:VIDeo:FRATe****Command/Query****DESCRIPTION**

The command sets the frame rate of the custom video trigger.

The query returns the current frame rate of the custom video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:FRATe <frate>

<frate>:= {25Hz|30Hz|50Hz|60Hz}

**QUERY SYNTAX**

:TRIGger:VIDeo:FRATe?

**RESPONSE FORMAT**

<frate>

<frate>:= {25Hz|30Hz|50Hz|60Hz}

**EXAMPLE**

The following command sets the frame rate of the custom video trigger to 50Hz.

Command message:

```
:TRIGger:VIDeo:FRATe 50Hz  
TRIG:VID:FRAT 50Hz
```

Query message:

```
TRIG:VID:FRAT?
```

Response message:

```
50Hz
```

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard

**:TRIGger:VIDeo:INTerlace****Command/Query****DESCRIPTION**

The command sets the interlace of the custom video trigger.

The query returns the current interlace of the custom video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:INTerlace <interlace>

<interlace>:= {1|2|4|8}

**QUERY SYNTAX**

:TRIGger:VIDeo:INTerlace?

**RESPONSE FORMAT**

<interlace>

<interlace>:= {1|2|4|8}

**EXAMPLE**

The following command sets the interlace of the custom video trigger to 8:1.

Command message:

```
:TRIGger:VIDeo:INTerlace 8  
TRIG:VID:INT 8
```

Query message:

```
TRIG:VID:INT?
```

Response message:

```
8
```

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard



**:TRIGger:VIDeo:LCNT**

**Command/Query**

**DESCRIPTION**

The command sets the lines of the custom video trigger.

The query returns the current of lines of the custom video trigger.

If the "Of Lines" is set to 800, the correct relationship between the interface, of fields, trigger line and trigger field is as follows:

| Of Lines | Interlace | Of Fields | Trigger Line | Trigger Field |
|----------|-----------|-----------|--------------|---------------|
| 800      | 1:1       | 1         | 800          | 1             |
| 800      | 2:1       | 1/2/4/8   | 400          | 1/1~2/1~4/1~8 |
| 800      | 4:1       | 1/2/4/8   | 300          | 1/1~2/1~4/1~8 |
| 800      | 8:1       | 1/2/4/8   | 100          | 1/1~2/1~4/1~8 |

**COMMAND SYNTAX**

:TRIGger:VIDeo:LCNT <line\_cnt>

<line\_cnt>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [300, 2000].

**QUERY SYNTAX**

:TRIGger:VIDeo:LCNT?

**RESPONSE FORMAT**

<line\_cnt>

<line\_cnt>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the lines of the custom video trigger to 500.

Command message:

*:TRIGger:VIDeo:LCNT 500*  
*TRIG:VID:LCNT 500*

Query message:

*TRIG:VID:LCNT?*

Response message:

*500*

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard

**:TRIGger:VIDeo:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the video trigger.

The query returns the current trigger level of the video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:LEVel <level\_value>

<level\_value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:VIDeo:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format

**EXAMPLE**

The following command sets the trigger level of the video trigger to 0.5 V.

Command message:

*:TRIGger:VIDeo:LEVel 5.00E-01*  
*TRIG:VID:LEV 5.00E-01*

Query message:

*TRIG:VID:LEV?*

Response message:

*5.00E-01*

**:TRIGger:VIDeo:LINE**

**Command/Query**

**DESCRIPTION**

The command sets the synchronous trigger line when the video standard is not custom.

The query returns the current synchronous trigger line when the video standard is not custom.

**COMMAND SYNTAX**

:TRIGger:VIDeo:LINE <line>

<line>:= Value in NR1 format, including an integer and no decimal point, like 1.

The following table shows the corresponding relations between line and field for all video standards(except for custom)

| Standard                   | Field 1   | Field 2  |
|----------------------------|-----------|----------|
| NTSC                       | [1, 263]  | [1, 262] |
| PAL                        | [1, 313]  | [1, 312] |
| HDTV 720P/50,<br>720P/60   | [1, 750]  |          |
| HDTV 1080P/50,<br>1080P/60 | [1, 1125] |          |
| HDTV 1080i/50,<br>1080i/60 | [1, 563]  | [1, 562] |

**QUERY SYNTAX**

:TRIGger:VIDeo:LINE?

**RESPONSE FORMAT**

<line>

<line>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the synchronous trigger line to 2.

Command message:

*:TRIGger:VIDeo:LINE 2*  
*TRIG:VID:LINE 2*

Query message:

*TRIG:VID:LINE?*

Response message:

*2*

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard  
:TRIGger:VIDeo:SYNC

**:TRIGger:VIDeo:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the video trigger.

The query returns the current trigger source of the video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:VIDeo:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the trigger source of the video trigger to channel 2.

Command message:

*:TRIGger:VIDeo:SOURce C2*

*TRIG:VID:SOUR C2*

Query message:

*TRIG:VID:SOUR?*

Response message:

*C2*

**:TRIGger:VIDeo:STANdard**

**Command/Query**

**DESCRIPTION**

The command sets the standard of the video trigger.

The query returns the current standard of the video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:STANdard <standard>

<standard>:=  
 {NTSC|PAL|P720L50|P720L60|P1080L50|P1080L60|I1080L50  
 |I1080L60|CUSTom}

**QUERY SYNTAX**

:TRIGger:VIDeo:STANdard?

**RESPONSE FORMAT**

<standard>

<standard>:=  
 {NTSC|PAL|P720L50|P720L60|P1080L50|P1080L60|I1080L50  
 |I1080L60|CUSTom}

**EXAMPLE**

The following command sets the standard of the video trigger to NTSC.

Command message:

*:TRIGger:VIDeo:STANdard NTSC  
 TRIG:VID:STAN NTSC*

Query message:

*TRIG:VID:STAN?*

Response message:

*NTSC*

**:TRIGger:VIDeo:SYNC****Command/Query****DESCRIPTION**

The command sets the sync mode of the video trigger.

The query returns the current sync mode of the video trigger.

**COMMAND SYNTAX**

:TRIGger:VIDeo:SYNC <sync>

<sync>:= {SELEct|ANY}

**QUERY SYNTAX**

:TRIGger:VIDeo:SYNC?

**RESPONSE FORMAT**

<sync>

<sync>:= {SELEct|ANY}

**EXAMPLE**

The following command sets the sync mode of the video trigger to select.

Command message:

```
:TRIGger:VIDeo:SYNC SELEct  
TRIG:VID:SYNC SEL
```

Query message:

```
TRIG:VID:SYNC?
```

Response message:

```
SELEct
```

**RELATED COMMANDS**

:TRIGger:VIDeo:STANdard

:TRIGger:VIDeo:LINE

:TRIGger:VIDeo:FIELd

## **:TRIGger:WINDow Commands**

The :TRIGGER:WINDow subsystem commands control the window trigger parameters.

- ◆ **:TRIGger:WINDow:CLEVel**
- ◆ **:TRIGger:WINDow:COUPling**
- ◆ **:TRIGger:WINDow:DLEVel**
- ◆ **:TRIGger:WINDow:HLDEVent**
- ◆ **:TRIGger:WINDow:HLTime**
- ◆ **:TRIGger:WINDow:HLEVel**
- ◆ **:TRIGger:WINDow:HOLDoff**
- ◆ **:TRIGger:WINDow:HStart**
- ◆ **:TRIGger:WINDow:LLEVel**
- ◆ **:TRIGger:WINDow:NREJect**
- ◆ **:TRIGger:WINDow:SOURce**
- ◆ **:TRIGger:WINDow:TYPE**

**:TRIGger:WINDow:CLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the center level of the window trigger.  
The query returns the current center level of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:CLEVel <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:WINDow:CLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the center level of the window trigger to 0.5 V.

Command message:

*:TRIGger:WINDow:CLEVel 5.00E-01*  
*TRIG:WIND:CLEV 5.00E-01*

Query message:

*TRIG:WIND:CLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:WINDow:DLEVel



**:TRIGger:WINDow:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the window trigger.

The query returns the current coupling mode of the window trigger

**COMMAND SYNTAX**

:TRIGger:WINDow:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a low-pass filter in the trigger path to remove high-frequency components from the trigger waveform. Use the high frequency rejection filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:WINDow:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets the coupling mode of the window trigger to DC.

Command message:

*:TRIGger:WINDow:COUPling DC*

*TRIG:WIND:COUP DC*

Query message:

*TRIG:WIND:COUP?*

Response message:

*DC*

**:TRIGger:WINDow:DLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the delta level of window trigger.

The query returns the current delta level of window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:DLEVel <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:WINDow:DLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the delta level of window trigger to 0.5 V.

Command message:

*:TRIGger:WINDow:DLEVel 5.00E-01*  
*TRIG:WIND:DLEV 5.00E-01*

Query message:

*TRIG:WIND:DLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:WINDow:CLEVel

**:TRIGger:WINDow:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the window trigger.

The query returns the current number of holdoff events of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:WINDow:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the window trigger to 3.

Command message:

*:TRIGger:WINDow:HLDEvent 3*  
*TRIG:WIND:HLDEV 3*

Query message:

*TRIG:WIND:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

:TRIGger:WINDow:HOLDoff

**:TRIGger:WINDow:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the window trigger.

The query returns the current holdoff time of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:HLDTIME <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A<br>SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [8.00E-09, 3.00E+01] |

**QUERY SYNTAX**

:TRIGger:WINDow:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the holdoff time of the window trigger to 15 ns.

Command message:

*:TRIGger:WINDow:HLDTIME 1.50E-08*  
*TRIG:WIND:HLDT 1.50E-08*

Query message:

*TRIG:WIND:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:WINDow:HOLDoff

**:TRIGger:WINDow:HLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the high trigger level of window trigger.

The query returns the current high trigger level of window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:HLEVel <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The high level value cannot be less than the low level value using by the command :TRIGger:WINDow:LLEVel.

**QUERY SYNTAX**

:TRIGger:WINDow:HLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the high trigger level of window trigger to 0.5 V.

Command message:

*:TRIGger:WINDow:HLEVel 5.00E-01*  
*TRIG:WIND:HLEV 5.00E-01*

Query message:

*TRIG:WIND:HLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:WINDow:LLEVel

**:TRIGger:WINDow:HOLDoff**

**Command/Query**

**DESCRIPTION**

The command selects the holdoff type of the window trigger.

The query returns the current holdoff type of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:WINDow:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type >:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the window trigger.

Command message:

*:TRIGger:WINDow:HOLDoff OFF*  
*TRIG:WIND:HOLD OFF*

Query message:

*TRIG:WIND:HOLD?*

Response message:

*OFF*

**RELATED COMMANDS**

:TRIGger:WINDow:HLDEvent  
 :TRIGger:WINDow:HLEVel  
 :TRIGger:WINDow:HStart

**:TRIGger:WINDow:HStart****Command/Query****DESCRIPTION**

The command defines the initial position of the window trigger holdoff.

The query returns the initial position of the window trigger holdoff.

**COMMAND SYNTAX**

:TRIGger:WINDow:HStart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:WINDow:HStart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to LAST\_TRIG (last trigger).

Command message:

*:TRIGger:WINDow:HStart LAST\_TRIG  
TRIG:WIND:HST LAST\_TRIG*

Query message:

*TRIG:WIND:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:WINDow:HOLDoff

**:TRIGger:WINDow:LLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the low trigger level of the window trigger.  
The query returns the current low trigger level of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:LLEVel <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The low level value cannot be greater than the high level value using by the command :TRIGger:WINDow:HLEVel.

**QUERY SYNTAX**

:TRIGger:WINDow:LLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the low trigger level of runt trigger to -0.5 V.

Command message:

*:TRIGger:WINDow:LLEVel -5.00E-01*  
*TRIG:WIND:LLEV -5.00E-01*

Query message:

*TRIG:WIND:LLEV?*

Response message:

*-5.00E-01*

**RELATED COMMANDS**

:TRIGger:WINDow:HLEVel



**:TRIGger:WINDow:NREJect****Command/Query****DESCRIPTION**

The command the state of noise reject.

The query returns the current state of noise reject.

**COMMAND SYNTAX**

:TRIGger:WINDow:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:WINDow:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the noise rejection.

Command message:

*:TRIGger:WINDow:NREJect ON*  
*TRIG:WIND:NREJ ON*

Query message:

*TRIG:WIND:NREJ?*

Response message:

*ON*

**:TRIGger:WINDow:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the window trigger.

The query returns the current trigger source of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:WINDow:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the trigger source of the window trigger to channel 2.

Command message:

*:TRIGger:WINDow:SOURce C2*

*TRIG:WIND:SOUR C2*

Query message:

*TRIG:WIND:SOUR?*

Response message:

*C2*

**:TRIGger:WINDow:TYPE****Command/Query****DESCRIPTION**

The command sets the window type of the window trigger.

The query returns the current window type of the window trigger.

**COMMAND SYNTAX**

:TRIGger:WINDow:TYPE <type>

<type>:= {ABSolute|RELative}

**QUERY SYNTAX**

:TRIGger:WINDow:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {ABSolute|RELative}

**EXAMPLE**

The following command sets the absolute type to window trigger.

Command message:

```
:TRIGger:WINDow:TYPE ABSolute  
TRIG:WIND:TYPE ABS
```

Query message:

```
TRIG:WIND:TYPE?
```

Response message:

```
ABSolute
```

**:TRIGger:INTerval Commands**

The :TRIGGER:INTerval subsystem commands control the interval trigger parameters.

- ◆ **:TRIGger:INTerval:COUPling**
- ◆ **:TRIGger:INTerval:HLDEvent**
- ◆ **:TRIGger:INTerval:HLDTIME**
- ◆ **:TRIGger:INTerval:HOLDoff**
- ◆ **:TRIGger:INTerval:HStart**
- ◆ **:TRIGger:INTerval:LEVel**
- ◆ **:TRIGger:INTerval:LIMit**
- ◆ **:TRIGger:INTerval:NREJect**
- ◆ **:TRIGger:INTerval:SLOPe**
- ◆ **:TRIGger:INTerval:SOURce**
- ◆ **:TRIGger:INTerval:TLOWer**
- ◆ **:TRIGger:INTerval:TUPPer**

**:TRIGger:INTerval:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the interval trigger.

The query returns the current coupling mode of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a 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.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:INTerval:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets the coupling mode of the interval trigger to DC.

Command message:

```
:TRIGger:INTerval:COUPling DC  
TRIG:INT:COUP DC
```

Query message:

```
TRIG:INT:COUP?
```

Response message:

```
DC
```

**:TRIGger:INTerval:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the interval trigger.

The query returns the current number of holdoff events of the interval trigger.

**COMMAND SYNTAX**

**:TRIGger:INTerval:HLDEvent <value>**

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

**:TRIGger:INTerval:HLDEvent?**

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the interval trigger to 3.

Command message:

*:TRIGger:INTerval:HLDEvent 3*

*TRIG:INT:HLDEV 3*

Query message:

*TRIG:INT:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

**:TRIGger:INTerval:HOLDoff**

**:TRIGger:INTerval:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the interval trigger.

The query returns the current holdoff time of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:HLDTIME <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model         | Value Range          |
|---------------|----------------------|
| SDS7000A      | [8.00E-09, 3.00E+01] |
| SDS5000X      |                      |
| SDS2000X Plus |                      |
| SDS6000 Pro   |                      |
| SDS6000A      |                      |
| SDS6000L      |                      |
| SDS3000X HD   |                      |
| SDS2000X HD   |                      |
| SDS1000X HD   |                      |
| SDS800X HD    |                      |

**QUERY SYNTAX**

:TRIGger:INTerval:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the holdoff time of the interval trigger to 15 ns.

Command message:

*:TRIGger:INTerval:HLDTIME 1.50E-08*

*TRIG:INT:HLDT 1.50E-08*

Query message:

*TRIG:INT:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:INTerval:HOLDoff

**:TRIGger:INTerval:HOLDoff**

**Command/Query**

**DESCRIPTION**

The command selects the holdoff type of the interval trigger.

The query returns the current holdoff type of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:INTerval:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type >:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the interval trigger.

Command message:

*:TRIGger:INTerval:HOLDoff OFF*  
*TRIG:INT:HOLD OFF*

Query message:

*TRIG:INT:HOLD?*

Response message:

*OFF*

**RELATED COMMANDS**

:TRIGger:INTerval:HLDEvent  
 :TRIGger:INTerval:HLTime  
 :TRIGger:INTerval:HStart



**:TRIGger:INTerval:HStart****Command/Query****DESCRIPTION**

The command sets the start holdoff mode of the interval trigger.

The query returns the current start holdoff mode of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:HStart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:INTerval:HStart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode of the interval trigger as LAST\_TRIG (last trigger).

Command message:

```
:TRIGger:INTerval:HStart LAST_TRIG  
TRIG:INT:HST LAST_TRIG
```

Query message:

```
TRIG:INT:HST?
```

Response message:

```
LAST_TRIG
```

**RELATED COMMANDS**

:TRIGger:INTerval:HOLDoff

**:TRIGger:INTerval:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the interval trigger.

The query returns the current trigger level of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:LEVel <level\_value>

<level\_value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:INTerval:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format

**EXAMPLE**

The following command sets the trigger level of the interval trigger to 0.5 V.

Command message:

*:TRIGger:INTerval:LEVel 5.00E-01*  
*TRIGr:INT:LEV 5.00E-01*

Query message:

*TRIG:INT:LEV?*

Response message:

*5.00E-01*

**:TRIGger:INTerval:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the interval trigger.

The query returns the current limit range type of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:INTerval:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the interval trigger to LESSthan.

Command message:

```
:TRIGger:INTerval:LIMit LESSthan  
TRIG:INT:LIM LESS
```

Query message:

```
TRIG:INT:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:TRIGger:INTerval:TLOWer

:TRIGger:INTerval:TUPPer

**:TRIGger:INTerval:NREJect****Command/Query****DESCRIPTION**

The command sets the state of the noise rejection.

The query returns the current state of the noise rejection function.

**COMMAND SYNTAX**

:TRIGger:INTerval:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:INTerval:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the noise rejection.

Command message:

*:TRIGger:INTerval:NREJect ON*

*TRIG:INT:NREJ ON*

Query message:

*TRIG:INT:NREJ?*

Response message:

*ON*

**:TRIGger:INTerval:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the interval trigger.

The query returns the current slope of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:INTerval:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the interval trigger.

Command message:

```
:TRIGger:INTerval:SLOPe RISing  
TRIG:INT:SLOP RIS
```

Query message:

```
TRIG:INT:SLOP?
```

Response message:

```
RISing
```

**:TRIGger:INTerval:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the interval trigger.

The query returns the current trigger source of the interval trigger.

**COMMAND SYNTAX**

:TRIGger:INTerval:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:INTerval:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the trigger source of the interval trigger as channel 1.

Command message:

*:TRIGger:INTerval:SOURce C1*

*TRIG:INT:SOUR C1*

Query message:

*TRIG:INT:SOUR?*

Response message:

*C1*

**:TRIGger:INTerval:TLOWer**  
**Command/Query**

**DESCRIPTION**

The command sets the lower value of the interval trigger limit type.

The query returns the current lower value of the interval trigger limit type.

**COMMAND SYNTAX**

:TRIGger:INTerval:TLOWer <value>

<value>:= Value in NR3 format. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:INTerval:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:INTerval:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format

**EXAMPLE**

The following command sets the time lower value of the interval trigger to 10 ns.

Command message:

*:TRIGger:INTerval:TLOWer 1.00E-08*  
*TRIG:INT:TLOW 1.00E-08*

Query message:

*TRIG:INT:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:INTerval:LIMit  
 :TRIGger:INTerval:TUPPer

**:TRIGger:INTerval:TUPPer****Command/Query****DESCRIPTION**

The command sets the upper value of the interval trigger limit type.

The query returns the current upper value of the interval trigger limit type.

**COMMAND SYNTAX**

:TRIGger:INTerval:TUPPer <value>

<value>:= Value in NR3 format. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:INTerval:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:INTerval:TUPPer?

**RESPONSE FORMAT**

<tupper\_value>

<tupper\_value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the time upper value of the interval trigger to 30 ns.

Command message:

```
:TRIGger:INTerval:TUPPer 3.00E-08
TRIG:INT:TUPP 3.00E-08
```

Query message:

```
TRIG:INT:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

```
:TRIGger:INTerval:LIMit
:TRIGger:INTerval:TLOWer
```



## **:TRIGger:DROPOut Commands**

The :TRIGGER:DROPOut subsystem commands control the dropout trigger parameters.

- ◆ **:TRIGger:DROPOut:COUPLing**
- ◆ **:TRIGger:DROPOut:HLDEvent**
- ◆ **:TRIGger:DROPOut:HLDTIME**
- ◆ **:TRIGger:DROPOut:HOLDoff**
- ◆ **:TRIGger:DROPOut:HStart**
- ◆ **:TRIGger:DROPOut:LEVel**
- ◆ **:TRIGger:DROPOut:NREJect**
- ◆ **:TRIGger:DROPOut:SLOPe**
- ◆ **:TRIGger:DROPOut:SOURce**
- ◆ **:TRIGger:DROPOut:TIME**
- ◆ **:TRIGger:DROPOut:TYPE**

**:TRIGger:DROPOut:COUPLing****Command/Query****DESCRIPTION**

The command sets the coupling mode of the dropout trigger.

The query returns the current coupling mode of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:COUPLing <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a low-pass filter in the trigger path to remove high-frequency components from the trigger waveform. Use the high-frequency rejection filter to remove high-frequency noise, such as AM or FM broadcast stations, from the trigger path.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:DROPOut:COUPLing?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets coupling mode of the dropout trigger to DC.

Command message:

```
:TRIGger:DROPOut:COUPLing DC
TRIG:DROP:COUP DC
```

Query message:

```
TRIG:DROP:COUP?
```

Response message:

```
DC
```

**:TRIGger:DROPOut:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the dropout trigger.

The query returns the current number of holdoff events of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:DROPOut:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the dropout trigger to 3.

Command message:

*:TRIGger:DROPOut:HLDEvent 3*  
*TRIG:DROP:HLDEV 3*

Query message:

*TRIG:DROP:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

:TRIGger:DROPOut:HOLDoff

**:TRIGger:DROPout:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the dropout trigger.  
 The query returns the current holdoff time of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPout:HLDTIME <value>

<value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model         | Value Range          |
|---------------|----------------------|
| SDS7000A      | [8.00E-09, 3.00E+01] |
| SDS5000X      |                      |
| SDS2000X Plus |                      |
| SDS6000 Pro   |                      |
| SDS6000A      |                      |
| SDS6000L      |                      |
| SDS3000X HD   |                      |
| SDS2000X HD   |                      |
| SDS1000X HD   |                      |
| SDS800X HD    |                      |

**QUERY SYNTAX**

:TRIGger:DROPout:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the holdoff time of the dropout trigger to 15 ns.

Command message:

*:TRIGger:DROPout:HLDTIME 1.50E-08*

*:TRIG:DROP:HLDT 1.50E-08*

Query message:

*TRIG:DROP:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:DROPout:HOLDoff

**:TRIGger:DROPOut:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the dropout trigger.

The query returns the current holdoff type of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:DROPOut:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type>:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the dropout trigger.

Command message:

*:TRIGger:DROPOut:HOLDoff OFF*  
*TRIG:DROP:HOLD OFF*

Query message:

*TRIG:DROP:HOLD?*

Response message:

*OFF*

**RELATED COMMANDS**

:TRIGger:DROPOut:HLDEvent  
:TRIGger:DROPOut:HLTime  
:TRIGger:DROPOut:HStart

**:TRIGger:DROPOut:HSTart****Command/Query****DESCRIPTION**

The command sets the start holdoff mode of the dropout trigger.

The query returns the current start holdoff mode of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:HSTart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:DROPOut:HSTart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start hold off mode to LAST\_TRIG (last trigger).

Command message:

```
:TRIGger:DROPOut:HSTart LAST_TRIG
TRIG:DROP:HST LAST_TRIG
```

Query message:

```
TRIG:DROP:HST?
```

Response message:

```
LAST_TRIG
```

**RELATED COMMANDS**

:TRIGger:DROPOut:HOLDoff

**:TRIGger:DROPOut:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the dropout trigger.

The query returns the current trigger level of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:LEVel <level\_value>

<level\_value>:= Value in NR3 format.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:DROPOut:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the trigger level of the dropout trigger to 0.5 V.

Command message:

*:TRIGger:DROPOut:LEVel 5.00E-1*  
*TRIG:DROP:LEV 5.00E-1*

Query message:

*TRIG:DROP:LEV?*

Response message:

*5.00E-01*

**:TRIGger:DROPOut:NREJect****Command/Query****DESCRIPTION**

The command sets the state of the noise rejection.

The query returns the current state of the noise rejection function.

**COMMAND SYNTAX**

:TRIGger:DROPOut:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:DROPOut:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the noise rejection.

Command message:

```
:TRIGger:DROPOut:NREJect ON  
TRIG:DROP:NREJ ON
```

Query message:

```
TRIG:DROP:NREJ?
```

Response message:

```
ON
```



**:TRIGger:DROPOut:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the dropout trigger.

The query returns the current slope of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:DROPOut:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the dropout trigger.

Command message:

```
:TRIGger:DROPOut:SLOPe RISing  
TRIG:DROP:SLOP RIS
```

Query message:

```
TRIG:DROP:SLOP?
```

Response message:

```
RISing
```

**:TRIGger:DROPOut:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the dropout trigger.

The query returns the current trigger source of the dropout trigger.

**COMMAND SYNTAX**

**:TRIGger:DROPOut:SOURce <source>**

**<source>:= {C<n>|D<d>}**

**<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.**

**<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.**

**QUERY SYNTAX**

**:TRIGger:DROPOut:SOURce?**

**RESPONSE FORMAT**

**<source>**

**<source>:= {C<n>|D<d>}**

**EXAMPLE**

The following command sets the trigger source of the dropout trigger to channel 2.

Command message:

*:TRIGger:DROPOut:SOURce C2*

*TRIG:DROP:SOUR C2*

Query message:

*TRIG:DROP:SOUR?*

Response message:

*C2*

**:TRIGger:DROPOut:TIME**

**Command/Query**

**DESCRIPTION**

The command sets the dropout time of the dropout trigger.

The query returns the current time of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:TIME <time>

<time>:= Value in NR3 format. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**QUERY SYNTAX**

:TRIGger:DROPOut:TIME?

**RESPONSE FORMAT**

<time>

<time>:= Value in NR3 format

**EXAMPLE**

The following command sets the time of the dropout trigger to 10 ns.

Command message:

*:TRIGger:DROPOut:TIME 1.00E-08*  
*TRIG:DROP:TIME 1.00E-08*

Query message:

*TRIG:DROP:TIME?*

Response message:

*1.00E-08*

**:TRIGger:DROPOut:TYPE****Command/Query****DESCRIPTION**

The command sets the over time type of the dropout trigger.

The query returns the current over time type of the dropout trigger.

**COMMAND SYNTAX**

:TRIGger:DROPOut:TYPE <type>

<type>:= {EDGE|STATE}

**QUERY SYNTAX**

:TRIGger:DROPOut:TYPE?

**RESPONSE FORMAT**

<type>

<type>:= {EDGE|STATE}

**EXAMPLE**

The following command sets the over time type of the dropout trigger to EDGE.

Command message:

```
:TRIGger:DROPOut:TYPE EDGE  
TRIG:DROP:TYPE EDGE
```

Query message:

```
TRIG:DROP:TYPE?
```

Response message:

```
EDGE
```

## **:TRIGger:RUNT Commands**

The :TRIGGER:RUNT subsystem commands control the runt trigger parameters.

- ◆ **:TRIGger:RUNT:COUPling**
- ◆ **:TRIGger:RUNT:HLDEvent**
- ◆ **:TRIGger:RUNT:HLDTIME**
- ◆ **:TRIGger:RUNT:HLEVel**
- ◆ **:TRIGger:RUNT:HOLDoff**
- ◆ **:TRIGger:RUNT:HStart**
- ◆ **:TRIGger:RUNT:LIMit**
- ◆ **:TRIGger:RUNT:LLEVel**
- ◆ **:TRIGger:RUNT:NREJect**
- ◆ **:TRIGger:RUNT:POLarity**
- ◆ **:TRIGger:RUNT:SOURce**
- ◆ **:TRIGger:RUNT:TLOWer**
- ◆ **:TRIGger:RUNT:TUPPer**

**:TRIGger:RUNT:COUPling****Command/Query****DESCRIPTION**

The command sets the coupling mode of the runt trigger.

The query returns the current coupling mode of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:COUPling <mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

- ◆ DC coupling allows dc and ac signals into the trigger path.
- ◆ AC coupling places a high-pass filter 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.
- ◆ HFREJect which is a high-frequency rejection filter adds a 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.
- ◆ LFREJect which is a low frequency rejection filter adds a 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.

**QUERY SYNTAX**

:TRIGger:RUNT:COUPling?

**RESPONSE FORMAT**

<mode>

<mode>:= {DC|AC|LFREJect|HFREJect}

**EXAMPLE**

The following command sets coupling mode of the runt trigger to DC.

Command message:

```
:TRIGger:RUNT:COUPling DC
TRIG:RUNT:COUP DC
```

Query message:

```
TRIG:RUNT:COUP?
```

Response message:

```
DC
```

**:TRIGger:RUNT:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the runt trigger.

The query returns the current number of holdoff events of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:RUNT:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the runt trigger to 3.

Command message:

*:TRIGger:RUNT:HLDEvent 3*  
*TRIG:RUNT:HLDEV 3*

Query message:

*TRIG:RUNT:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

:TRIGger:RUNT:HOLDoff

**:TRIGger:RUNT:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the runt trigger.

The query returns the current holdoff time of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model         | Value Range          |
|---------------|----------------------|
| SDS7000A      | [8.00E-09, 3.00E+01] |
| SDS5000X      |                      |
| SDS2000X Plus |                      |
| SDS6000 Pro   |                      |
| SDS6000A      |                      |
| SDS6000L      |                      |
| SDS3000X HD   |                      |
| SDS2000X HD   |                      |
| SDS1000X HD   |                      |
| SDS800X HD    |                      |

**QUERY SYNTAX**

:TRIGger:RUNT:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the holdoff time of the runt trigger to 15 ns.

Command message:

*:TRIGger:RUNT:HLDTIME 1.50E-08*  
*TRIG:RUNT:HLDT 1.50E-08*

Query message:

*TRIG:RUNT:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:DROPout:HOLDoff



**:TRIGger:RUNT:HLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the high trigger level of the runt trigger.

The query returns the current high trigger level of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:HLEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**Note:**

The high level value cannot be less than the low level value using by the command :TRIGger:RUNT:LLEVel.

**QUERY SYNTAX**

:TRIGger:RUNT:HLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the high trigger level of the runt trigger to 0.5 V.

Command message:

*:TRIGger:RUNT:HLEVel 5.00E-01*  
*TRIG:RUNT:HLEV 5.00E-01*

Query message:

*TRIG:RUNT:HLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:RUNT:LLEVel

**:TRIGger:RUNT:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the runt trigger.

The query returns the current holdoff type of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:RUNT:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type>:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the runt trigger.

Command message:

```
:TRIGger:RUNT:HOLDoff OFF
TRIG:RUNT:HOLD OFF
```

Query message:

```
TRIG:RUNT:HOLD?
```

Response message:

```
OFF
```

**RELATED COMMANDS**

```
:TRIGger:RUNT:HLDEvent
:TRIGger:RUNT:HLDTime
:TRIGger:RUNT:HStart
```

**:TRIGger:RUNT:HStart****Command/Query****DESCRIPTION**

The command sets the start holdoff mode of the runt trigger.

The query returns the current start holdoff mode of the runt trigger.

**COMMAND SYNTAX**

:TRIGger: RUNT:HStart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:RUNT:HStart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to LAST\_TRIG (last trigger).

Command message:

*:TRIGger:RUNT:HStart LAST\_TRIG  
TRIG:RUNT:HST LAST\_TRIG*

Query message:

*TRIG:RUNT:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:RUNT:HOLDoff

**:TRIGger:RUNT:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the runt trigger.

The query returns the current limit range type of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:RUNT:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the runt trigger to LESSthan.

Command message:

```
:TRIGger:RUNT:LIMit LESSthan  
TRIG:RUNT:LIM LESS
```

Query message:

```
TRIG:RUNT:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

```
:TRIGger:RUNT:TLOWer  
:TRIGger:RUNT:TUPPer
```

**:TRIGger:RUNT:LLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the low trigger level of the runt trigger.

The query returns the current low trigger level of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:LLEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$   |

**Note:**

The low level value cannot be greater than the high level value using by the command :TRIGger:RUNT:HLEVel.

**QUERY SYNTAX**

:TRIGger:RUNT:LLEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the low trigger level of the runt trigger to -0.5 V.

Command message:

*:TRIGger:RUNT:LLEVel -5.00E-01*  
*TRIG:RUNT:LLEV -5.00E-01*

Query message:

*TRIG:RUNT:LLEV?*

Response message:

*-5.00E-01*

**RELATED COMMANDS**

:TRIGger:RUNT:HLEVel

**:TRIGger:RUNT:NREJect****Command/Query****DESCRIPTION**

The command sets the state of noise rejection.

The query returns the current state of noise rejection function.

**COMMAND SYNTAX**

:TRIGger:RUNT:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:RUNT:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on the noise rejection.

Command message:

*:TRIGger:RUNT:NREJect ON*  
*TRIG:RUNT:NREJ ON*

Query message:

*TRIG:RUNT:NREJ?*

Response message:

*ON*

**:TRIGger:RUNT:POLarity****Command/Query****DESCRIPTION**

The command sets the polarity of the runt trigger.

The query returns the current polarity of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:POLarity <polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**QUERY SYNTAX**

:TRIGger:RUNT:POLarity?

**RESPONSE FORMAT**

<polarity\_type>

<polarity\_type>:= {POSitive|NEGative}

**EXAMPLE**

The following command sets the polarity of the runt trigger to POSitive.

Command message:

*:TRIGger:RUNT:POLarity POSitive*  
*TRIG:RUNT:POL POS*

Query message:

*TRIG:RUNT:POL?*

Response message:

*POSitive*

**:TRIGger:RUNT:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the runt trigger.

The query returns the current trigger source of the runt trigger.

**COMMAND SYNTAX**

:TRIGger:RUNT:SOURce <source>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:RUNT:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>}

**EXAMPLE**

The following command sets the trigger source of the runt trigger to channel 2

Command message:

```
:TRIGger:RUNT:SOURce C2  
TRIG:RUNT:SOUR C2
```

Query message:

```
TRIG:RUNT:SOUR?
```

Response message:

```
C2
```



**:TRIGger:RUNT:TLOWer**  
**Command/Query**

**DESCRIPTION**

The command sets the lower value of the runt trigger limit type.

The query returns the current lower value of the runt trigger limit type.

**COMMAND SYNTAX**

:TRIGger:RUNT:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS7000A<br>SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:RUNT:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:RUNT:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the lower time of the runt trigger to 10 ns.

Command message:

*:TRIGger:RUNT:TLOWer 1.00E-08*  
*TRIG:RUNT:TLOW 1.00E-08*

Query message:

*TRIG:RUNT:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:RUNT:TUPPer  
 :TRIGger:RUNT:LIMit

**:TRIGger:RUNT:TUPPer****Command/Query****DESCRIPTION**

The command sets the upper value of the runt trigger limit type.

The query returns the current upper value of the runt trigger limit type.

**COMMAND SYNTAX**

:TRIGger:PULse:RUNT <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value varies by model, see the table below for details.

| Model   | Value Range          |
|---|----------------------|
| SDS7000A  | [1.00E-09, 2.00E+01] |
| SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [2.00E-09, 2.00E+01] |
| SHS800X/SHS1000X  | [2.00E-09, 4.20E+00] |

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:RUNT:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:RUNT:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the upper time of the runt trigger to 30 ns.

Command message:

```
:TRIGger:RUNT:TUPPer 3.00E-08
TRIG:RUNT:TUPP 3.00E-08
```

Query message:

```
TRIG:RUNT:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

```
:TRIGger:RUNT:LIMit
:TRIGger:RUNT:TLOWer
```

## **:TRIGger:PATtern Commands**

The :TRIGGER:PATTERN subsystem commands control the pattern trigger parameters.

- ◆ **:TRIGger:PATtern:HLDEvent**
- ◆ **:TRIGger:PATtern:HLTime**
- ◆ **:TRIGger:PATtern:HOLDoff**
- ◆ **:TRIGger:PATtern:HStart**
- ◆ **:TRIGger:PATtern:INPut**
- ◆ **:TRIGger:PATtern:LEVel**
- ◆ **:TRIGger:PATtern:LIMit**
- ◆ **:TRIGger:PATtern:LOGic**
- ◆ **:TRIGger:PATtern:TLOWer**
- ◆ **:TRIGger:PATtern:TUPPer**

**:TRIGger:PATtern:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the pattern trigger.

The query returns the current number of holdoff events of the pattern trigger.

**COMMAND SYNTAX**

**:TRIGger:PATtern:HLDEvent <value>**

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

**:TRIGger:PATtern:HLDEvent?**

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the pattern trigger to 3.

Command message:

*:TRIGger:PATtern:HLDEvent 3*  
*TRIG:PATT:HLDEV 3*

Query message:

*TRIG:PATT:HLDEV?*

Response message:

*3*

**RELATED COMMANDS**

**:TRIGger:PATtern:HOLDoff**

**:TRIGger:PATtern:HLDTIME**

**Command/Query**

**DESCRIPTION**

This This command sets the holdoff time of the pattern trigger.

The query returns the current holdoff time of the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Mode  | value                 |
|---|-----------------------|
| SDS7000A<br>SDS5000X<br>SDS2000X Plus<br>SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [8.00E-09, 3.00E+01]  |
| SHS800X/SHS1000X  | [80.00E-09, 1.50E+00] |

**QUERY SYNTAX**

:TRIGger:PATtern:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the holdoff time of the pattern trigger to 15 ns.

Command message:

*:TRIGger:PATtern:HLDTIME 1.50E-08*  
*TRIG:PATT:HLDT 1.50E-08*

Query message:

*TRIG:PATT:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:PATtern:HOLDoff

**:TRIGger:PATtern:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the pattern trigger.

The query returns the current holdoff type of the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff
- ◆ EVENTs means the amount of events that the oscilloscope counts before re-arming the trigger circuitry
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry

**QUERY SYNTAX**

:TRIGger:PATtern:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

< holdoff\_type >:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the pattern trigger.

Command message:

```
:TRIGger:PATtern:HOLDoff OFF  
TRIG:PATT:HOLD OFF
```

Query message:

```
TRIG:PATT:HOLD?
```

Response message:

```
OFF
```

**RELATED COMMANDS**

```
:TRIGger:PATtern:HLDEvent  
:TRIGger:PATtern:HLTime  
:TRIGger:PATtern:HStart
```

**:TRIGger:PATtern:HStart****Command/Query****DESCRIPTION**

The command sets the start holdoff mode of the pattern trigger.

The query returns the current start holdoff mode of the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:HStart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:PATtern:HStart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to LAST\_TRIG (last trigger).

Command message:

*:TRIGger:PATtern:HStart LAST\_TRIG  
TRIG:PATT:HST LAST\_TRIG*

Query message:

*TRIG:PATT:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:PATtern:HOLDoff

**:TRIGger:PATtern:INPut**

**Command/Query**

**DESCRIPTION**

The command specifies the logical input condition for the channel (Cx) and digital channel (Dx) of the pattern trigger.

The query returns the logical input condition of pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:INPut <logic>[...[,<logic>]]

<logic>:= {X|L|H}

- ◆ X means the "don't care" state.
- ◆ H means the logic high state.
- ◆ L means the logic low state.

**Note:**

Parameters are configured to corresponding sources in the order of C1-C<n>, D0-D15.

**QUERY SYNTAX**

:TRIGger:PATtern:INPut?

**RESPONSE FORMAT**

<input>

<input>:= {X|L|H}

**EXAMPLE**

The following command sets the logic input for channel 1 to H, for channel 2 to H, for channel 3 to L, for channel 4 to X and for all digital channel to X.

Command message:

```
:TRIGger:PATtern:INPut
H,H,L,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X
TRIG:PATT:INP H,H,L,X,X,X,X,X,X,X,X,X,X,X,X,X
```

Query message:

```
TRIG:PATT:INP?
```

Response message:

```
H,H,L,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X
```



**:TRIGger:PATtern:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of source in the pattern trigger.

The query returns the current trigger level of source in the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:LEVel <source>,<value>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:PATtern:LEVel? <source>

**RESPONSE FORMAT**

<source>,<value>

<source>:= {C<n>}

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the pattern trigger level to 0.5 V.

Command message:

*:TRIGger:PATtern:LEVel C2,5.00E-01*  
*TRIG:PATT:LEV C2,5.00E-01*

Query message:

*TRIG:PATT:LEV? C2*

Response message:

*C2,5.00E-01*

**RELATED COMMANDS**

:TRIGger:PATtern:INPut

**:TRIGger:PATtern:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the pattern trigger.

The query returns the current limit range type of the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:PATtern:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of pattern trigger to LESSthan.

Command message:

```
:TRIGger:PATtern:LIMit LESSthan  
TRIG:PATT:LIM LESS
```

Query message:

```
TRIG:PATT:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:TRIGger:PATtern:TLOWer

:TRIGger:PATtern:TUPPer

**:TRIGger:PATtern:LOGic****Command/Query****DESCRIPTION**

The command sets the logical combination of the input channels for the pattern trigger.

The query returns the current logical combination of the pattern trigger.

**COMMAND SYNTAX**

:TRIGger:PATtern:LOGic <type>

<type>:= {AND|OR|NAND|NOR}

**QUERY SYNTAX**

:TRIGger:PATtern:LOGic?

**RESPONSE FORMAT**

<logic\_type>

<logic\_type>:= {AND|OR|NAND|NOR}

**EXAMPLE**

The following command sets the logic mode of the pattern trigger to AND.

Command message:

*:TRIGger:PATtern:LOGic AND  
TRIG:PATT:LOG AND*

Query message:

*TRIG:PATT:LOG?*

Response message:

*AND*

**:TRIGger:PATtern:TLOWer****Command/Query****DESCRIPTION**

The command sets the lower value of the pattern trigger limit type.

The query returns the current lower value of the pattern trigger limit type.

**COMMAND SYNTAX**

:TRIGger:PATtern:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [2.00E-09, 2.00E+01].

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:PATtern:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:PATtern:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the lower time of the pattern trigger to 10 ns.

Command message:

```
:TRIGger:PATtern:TLOWer 1.00E-08  
TRIG:PATT:TLOW 1.00E-08
```

Query message:

```
TRIG:PATT:TLOW?
```

Response message:

```
1.00E-08
```

**RELATED COMMANDS**

:TRIGger:PATtern:LIMit  
:TRIGger:PATtern:TUPPer

**:TRIGger:PATtern:TUPPer****Command/Query****DESCRIPTION**

The command sets the upper value of the pattern trigger limit type.

The query returns the current upper value of the pattern trigger limit type.

**COMMAND SYNTAX**

:TRIGger:PATtern:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [3.00E-09, 2.00E+01].

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:PATtern:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:PATtern:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the upper time of the pattern trigger to 30 ns.

Command message:

```
:TRIGger:PATtern:TUPPer 3.00E-08  
TRIG:PATT:TUPP 3.00E-08
```

Query message:

```
TRIG:PATT:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

:TRIGger:PATtern:LIMit  
:TRIGger:PATtern:TLOWer

**:TRIGger:QUALified Commands**

The :TRIGGER:QUALified subsystem commands control the qualified trigger parameters.

- ◆ **:TRIGger:QUALified:ELEVel**
- ◆ **:TRIGger:QUALified:ESLope**
- ◆ **:TRIGger:QUALified:ESource**
- ◆ **:TRIGger:QUALified:LIMit**
- ◆ **:TRIGger:QUALified:QLEVel**
- ◆ **:TRIGger:QUALified:QSource**
- ◆ **:TRIGger:QUALified:TLOWer**
- ◆ **:TRIGger:QUALified:TUPPer**
- ◆ **:TRIGger:QUALified:TYPE**

**:TRIGger:QUALified:ELEVel**

**Command/Query**

**DESCRIPTION**

The command sets the edge trigger level of the edge source in the qualified trigger.

The query returns the current edge trigger level in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:ELEVel <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L  | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:QUALified:ELEVel?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the edge trigger level value of the qualified trigger to 0.5 V.

Command message:

*:TRIGger:QUALified:ELEVel 5.00E-01*  
*TRIG:QUAL:ELEV 5.00E-01*

Query message:

*TRIG:QUAL:ELEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:QUALified:QLEVel

**:TRIGger:QUALified:ESLope****Command/Query****DESCRIPTION**

The command sets the edge trigger slope in the qualified trigger.

The query returns the current edge trigger slope in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:ESLope <type>

<type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:QUALified:ESLope?

**RESPONSE FORMAT**

<type>

<type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the edge trigger slope in the qualified trigger to RISing.

Command message:

*:TRIGger:QUALified:ESLope RISing*

*TRIG:QUAL:ESL RIS*

Query message:

*TRIG:QUAL:ESL?*

Response message:

*RISing*

**RELATED COMMANDS**

:TRIGger:QUALified:TYPE



**:TRIGger:QUALified:ESource****Command/Query****DESCRIPTION**

The command sets the edge trigger source in the qualified trigger.

The query returns the current edge trigger source in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:ESource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:QUALified:ESource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the edge trigger source of the qualified trigger to channel 1.

Command message:

*:TRIGger:QUALified:ESource C1*

*TRIG:QUAL:ES C1*

Query message:

*TRIG:QUAL:ES?*

Response message:

*C1*

**RELATED COMMANDS**

:TRIGger:QUALified:QSource

**:TRIGger:QUALified:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type when the qualified type is “State with Delay” or “Edge with Delay” in the qualified trigger.

The query returns the current limit range type in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:QUALified:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit range type to LESSthan in the qualified trigger.

Command message:

```
:TRIGger:QUALified:LIMit LESSthan  
TRIG:QUAL:LIM LESS
```

Query message:

```
TRIG:QUAL:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:TRIGger:QUALified:TLOWer

:TRIGger:QUALified:TUPPer

**:TRIGger:QUALified:QLEVel**

**Command/Query**

**DESCRIPTION**

The command sets the level of the qualify source in the qualified trigger.

The query returns the current level of the qualify source in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:QLEVel <level>

<level>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L  | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:QUALified:QLEVel?

**RESPONSE FORMAT**

<level>

<level>:= Value in NR3 format.

**EXAMPLE**

The following command sets the level of the qualify source in the qualified trigger to 0.5 V.

Command message:

*:TRIGger:QUALified:QLEVel 5.00E-01*  
*TRIG:QUAL:QLEV 5.00E-01*

Query message:

*TRIG:QUAL:QLEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:QUALified:ELEVel

**:TRIGger:QUALified:QSource****Command/Query****DESCRIPTION**

The command sets the qualify source of the qualified trigger.

The query returns the current qualify source of the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:QSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:QUALified:QSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the qualify source of the qualified trigger as channel 1.

Command message:

*:TRIGger:QUALified:QSource C1*

*TRIG:QUAL:QS C1*

Query message:

*TRIG:QUAL:QS?*

Response message:

*C1*

**RELATED COMMANDS**

:TRIGger:QUALified:ESource

**:TRIGger:QUALified:TLOWer****Command/Query****DESCRIPTION**

The command sets the limit lower value when the qualified type is “Edge with Delay” or “State with Delay” in the qualified trigger.

The query returns the current delay lower value in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [2.00E-09, 2.00E+01].

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:QUALified:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:QUALified:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the lower time of the qualified trigger to 10 ns.

Command message:

```
:TRIGger:QUALified:TLOWer 1.00E-08  
TRIG:QUAL:TLOW 1.00E-08
```

Query message:

```
TRIG:QUAL:TLOW?
```

Response message:

```
1.00E-08
```

**RELATED COMMANDS**

:TRIGger:QUALified:LIMit  
:TRIGger:QUALified:TUPPer

**:TRIGger:QUALified:TUPPer****Command/Query****DESCRIPTION**

The command sets limit upper value when the qualified type is “Edge with Delay” or “State with Delay” in the qualified trigger.

The query returns the current delay upper value in the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [3.00E-09, 2.00E+01].

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:QUALified:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:QUALified:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the delay upper value of the qualified trigger to 30 ns.

Command message:

```
:TRIGger:QUALified:TUPPer 3.00E-08  
TRIG:QUAL:TUPP 3.00E-08
```

Query message:

```
TRIG:QUAL:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

:TRIGger:QUALified:LIMit  
:TRIGger:QUALified:TLOWer

**:TRIGger:QUALified:TYPE****Command/Query****DESCRIPTION**

The command sets the qualified type of the qualified trigger.

The query returns the current qualified type of the qualified trigger.

**COMMAND SYNTAX**

:TRIGger:QUALified:TYPE <type>[,<option>]

<type>:= {STATE|STATE\_DLY|EDGE|EDGE\_DLY}

<option>:= {LOW|HIGH} when <type> is STATE or STATE\_DLY

<option>:= {RISing|FALLing} when <type> is EDGE or EDGE\_DLY

**QUERY SYNTAX**

:TRIGger:QUALified:TYPE?

**RESPONSE FORMAT**

<type>[,<option>]

<type>:= {STATE|STATE\_DLY|EDGE|EDGE\_DLY}

<option>:= {LOW|HIGH} when <type> is STATE or STATE\_DLY

<option>:= {RISing|FALLing} when <type> is EDGE or EDGE\_DLY

**EXAMPLE**

The following command sets the qualified type of the qualified trigger to edge.

Command message:

*:TRIGger:QUALified:TYPE EDGE*  
*TRIG:QUAL:TYPE EDGE*

Query message:

*TRIG:QUAL:TYPE?*

Response message:

*EDGE*

**:TRIGger:DElay Commands**

The :TRIGGER:DElay subsystem commands control the delay trigger parameters.

- ◆ **:TRIGger:DElay:SOURce**
- ◆ **:TRIGger:DElay:SOURce2**
- ◆ **:TRIGger:DElay:SLOPe**
- ◆ **:TRIGger:DElay:SLOPe2**
- ◆ **:TRIGger:DElay:LEVel**
- ◆ **:TRIGger:DElay:LEVel2**
- ◆ **:TRIGger:DElay:LIMit**
- ◆ **:TRIGger:DElay:TUPPer**
- ◆ **:TRIGger:DElay:TLOWer**



**:TRIGger:DELay:SOURce**  
**Command/Query**

**DESCRIPTION**

The command sets the level state of trigger source A in the delay trigger.

The query returns the current level state of trigger source A in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DELay:SOURce <state>[...[, <state>]]

<state>:= {X|L|H}

- ◆ X means the "don't care" state.
- ◆ H means the logic high state.
- ◆ L means the logic low state.

**Note:**

Parameters are configured to corresponding sources in the order of C1-C<n>, D0-D15.

**QUERY SYNTAX**

:TRIGger:DELay:SOURce?

**RESPONSE FORMAT**

<state>

<state>:= {X|L|H}

**EXAMPLE**

The following command sets the logic input for channel 1 to H, for channel 2 to L, for channel 3 to L, for channel 4 to X and for all digital channel to X.

Command message:

```
:TRIGger:DELay:SOURce
H,L,L,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X
TRIG:DEL:SOUR H,L,L,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X
```

Query message:

```
TRIG:DEL:SOUR?
```

Response message:

```
H,L,L,X,X,X,X,X,X,X,X,X,X,X,X,X,X,X
```

**RELATED COMMANDS**

:TRIGger:DELay:SOURce2

**:TRIGger:DELay:SOURce2****Command/Query****DESCRIPTION**

The command sets the trigger source B in the delay trigger.

The query returns the current trigger source B in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DELay:SOURce2 <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:DELay:SOURce2?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the trigger source of source B in the delay trigger to channel 2.

Command message:

*:TRIGger:DELay:SOURce2 C2*

*TRIG:DEL:SOUR2 C2*

Query message:

*TRIG:DEL:SOUR2?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:DELay:SOURce

**:TRIGger:DELay:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of source A in the delay trigger.

The query returns the slope of source A in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DELay:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:DELay:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of source A in the delay trigger.

Command message:

```
:TRIGger:DELay:SLOPe RISing  
TRIG:DEL:SLOP RIS
```

Query message:

```
TRIG:DEL:SLOP?
```

Response message:

```
RISing
```

**:TRIGger:DElay:SLOPe2****Command/Query****DESCRIPTION**

The command sets the slope of source B in the delay trigger.

The query returns the slope of source B in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DElay:SLOPe2 <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:DElay:SLOPe2?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of source B in the delay trigger.

Command message:

```
:TRIGger:DElay:SLOPe2 RISing  
TRIG:DEL:SLOP2 RIS
```

Query message:

```
TRIG:DEL:SLOP2?
```

Response message:

```
RISing
```

**:TRIGger:DELay:LEVel****Command/Query****DESCRIPTION**

The command sets the level of source A in the delay trigger.

The query returns the current trigger level of source A in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DELay:LEVel <source>,<value>

<source>:= {C<n>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 * \text{vertical\_scale} - \text{vertical\_offset}, 4.26 * \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                 | $[-4.5 * \text{vertical\_scale} - \text{vertical\_offset}, 4.5 * \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 * \text{vertical\_scale} - \text{vertical\_offset}, 4.1 * \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:DELay:LEVel? <source>

<source>:= {C<n>}

**RESPONSE FORMAT**

<source>,<value>

<source>:= {C<n>}

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets delay trigger source A to C2 and level to 0.5 V

Command message:

```
:TRIGger:DELay:LEVel C2,5.00E-01
TRIG:DEL:LEV C2,5.00E-01
```

Query message:

```
TRIG:DEL:LEV? C2
```

Response message:

```
C2,5.00E-01
```

**RELATED COMMANDS**

:TRIGger:DELay:LEVel2

**:TRIGger:DELay:LEVel2**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of source B in the delay trigger.

The query returns the current trigger level of source B in the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DELay:LEVel2 <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X H<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:DELay:LEVel2?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets source B level of the delay trigger to 0.5 V

Command message:

*:TRIGger:DELay:LEVel2 5.00E-01*  
*TRIG:DEL:LEV2 5.00E-01*

Query message:

*TRIG:DEL:LEV2?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:DELay:LEVel

**:TRIGger:DElay:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the delay trigger.

The query returns the current limit range type of the delay trigger.

**COMMAND SYNTAX**

:TRIGger:DElay:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:DElay:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the delay trigger to LESSthan.

Command message:

```
:TRIGger:DElay:LIMit LESSthan  
TRIG:DEL:LIM LESS
```

Query message:

```
TRIG:DEL:LIM?
```

Response message:

```
LESSthan
```

**RELATED COMMANDS**

:TRIGger:DElay:TLOWer

:TRIGger:DElay:TUPPer

**:TRIGger:DELAy:TUPPer****Command/Query****DESCRIPTION**

The command sets the limit upper value of the delay trigger limit type.

The query returns the current limit upper value of the delay trigger limit type.

**COMMAND SYNTAX**

:TRIGger:DELAy:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [3.00E-09, 2.00E+01].

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:DELAy:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:DELAy:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper value of the delay trigger to 30 ns, when the limit range type is OUTer.

Command message:

```
:TRIGger:DELAy:TUPPer 3.00E-08  
TRIG:DEL:TUPP 3.00E-08
```

Query message:

```
TRIG:DEL:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

:TRIGger:DELAy:LIMit  
:TRIGger:DELAy:TLOWer



**:TRIGger:DElay:TLOWer**

**Command/Query**

**DESCRIPTION**

The command sets the limit lower value of the delay trigger limit type.

The query returns the current limit lower value of the delay trigger limit type.

**COMMAND SYNTAX**

:TRIGger:DElay:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [2.00E-09, 2.00E+01].

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:DElay:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:DElay:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the delay trigger to 10 ns.

Command message:

*:TRIGger:DElay:TLOWer 1.00E-08*  
*TRIG:DEL:TLOW 1.00E-08*

Query message:

*TRIG:DEL:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:DElay:LIMit  
 :TRIGger:DElay:TUPPer

**:TRIGger:NEDGE Commands**

The:TRIGGER:NEDGE subsystem commands control the Nth Edge trigger parameters.

- ◆ **:TRIGger:NEDGE:SOURce**
- ◆ **:TRIGger:NEDGE:SLOPe**
- ◆ **:TRIGger:NEDGE:IDLE**
- ◆ **:TRIGger:NEDGE:EDGE**
- ◆ **:TRIGger:NEDGE:LEVel**
- ◆ **:TRIGger:NEDGE:HOLDoff**
- ◆ **:TRIGger:NEDGE:HLTime**
- ◆ **:TRIGger:NEDGE:HLDEvent**
- ◆ **:TRIGger:NEDGE:HStart**
- ◆ **:TRIGger:NEDGE:NREJect**

**:TRIGger:NEDGe:SOURce****Command/Query****DESCRIPTION**

The command sets the trigger source of the Nth edge trigger.

The query returns the current trigger source of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:NEDGe:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the trigger source of the Nth edge trigger as C1.

Command message:

*:TRIGger:NEDGe:SOURce C1*

*TRIG:NEDG:SOUR C1*

Query message:

*TRIG:NEDG:SOUR?*

Response message:

*C1*

**RELATED COMMANDS**

:TRIGger:NEDGe:LEVel

**:TRIGger:NEDGe:SLOPe****Command/Query****DESCRIPTION**

The command sets the slope of the Nth edge trigger.

The query returns the current slope setting of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:NEDGe:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the rising slope of the Nth edge trigger.

Command message:

```
:TRIGger:NEDGe:SLOPe RISing  
TRIG:NEDG:SLOP RIS
```

Query message:

```
TRIG:NEDG:SLOP?
```

Response message:

```
RISing
```

**:TRIGger:NEDGe:IDLE**

**Command/Query**

**DESCRIPTION**

The command sets the idle time of the Nth edge trigger.

The query returns the current idle time of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:IDLE <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model       | Value Range          |
|-------------|----------------------|
| SDS7000A    | [8.00E-09, 2.00E+01] |
| SDS5000X    |                      |
| SDS6000 Pro |                      |
| SDS6000A    |                      |
| SDS6000L    |                      |
| SDS3000X HD |                      |
| SDS2000X HD |                      |
| SDS1000X HD |                      |
| SDS800X HD  |                      |

**QUERY SYNTAX**

:TRIGger:NEDGe:IDLE?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the idle time of the Nth edge trigger to 15 ns.

Command message:

*:TRIGger:NEDGe:IDLE 1.50E-08*

*TRIG:NEDG:IDLE 1.50E-08*

Query message:

*TRIG:NEDG:IDLE?*

Response message:

*1.50E-08*

**:TRIGger:NEDGe:EDGE****Command/Query****DESCRIPTION**

This command sets the edge num of the Nth edge trigger.

The query returns the current edge num of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:EDGE <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 65535].

**QUERY SYNTAX**

:TRIGger:NEDGe:EDGE?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the edge num of the Nth edge trigger to 3.

Command message:

```
:TRIGger:NEDGe:EDGE 3  
TRIG:NEDG:EDGE 3
```

Query message:

```
TRIG:NEDG:EDGE?
```

Response message:

```
3
```

**:TRIGger:NEDGe:LEVel**

**Command/Query**

**DESCRIPTION**

The command sets the trigger level of the Nth edge trigger.

The query returns the current trigger level value of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:LEVel <level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                 | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:NEDGe:LEVel?

**RESPONSE FORMAT**

<level\_value>

<level\_value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the trigger level of the Nth edge trigger to 0.5 V.

Command message:

*:TRIGger:NEDGe:LEVel 5.00E-01*  
*TRIG:NEDG:LEV 5.00E-01*

Query message:

*TRIG:NEDG:LEV?*

Response message:

*5.00E-01*

**RELATED COMMANDS**

:TRIGger:NEDGe:SOURce

**:TRIGger:NEDGe:HOLDoff****Command/Query****DESCRIPTION**

The command selects the holdoff type of the Nth edge trigger.

The query returns the current holdoff type of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:HOLDoff <holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

- ◆ OFF means to turn off the holdoff.
- ◆ EVENTs means the number of trigger events that the oscilloscope counts before re-arming the trigger circuitry.
- ◆ TIME means the amount of time that the oscilloscope waits before re-arming the trigger circuitry.

**QUERY SYNTAX**

:TRIGger:NEDGe:HOLDoff?

**RESPONSE FORMAT**

<holdoff\_type>

<holdoff\_type>:= {OFF|EVENTs|TIME}

**EXAMPLE**

The following command turns off the holdoff of the Nth edge trigger.

Command message:

```
:TRIGger:NEDGe:HOLDoff OFF
TRIG:NEDG:HOLD OFF
```

Query message:

```
TRIG:NEDG:HOLD?
```

Response message:

```
OFF
```

**RELATED COMMANDS**

```
:TRIGger:NEDGe:HLDEvent
:TRIGger:NEDGe:HLDTime
:TRIGger:NEDGe:HSTart
```



**:TRIGger:NEDGe:HLDTIME**

**Command/Query**

**DESCRIPTION**

The command sets the holdoff time of the Nth edge trigger.

The query returns the current holdoff time of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:HLDTIME <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model       | Value Range          |
|-------------|----------------------|
| SDS7000A    | [8.00E-09, 3.00E+01] |
| SDS5000X    |                      |
| SDS6000 Pro |                      |
| SDS6000A    |                      |
| SDS6000L    |                      |
| SDS3000X HD |                      |
| SDS2000X HD |                      |
| SDS1000X HD |                      |
| SDS800X HD  |                      |

**QUERY SYNTAX**

:TRIGger:NEDGe:HLDTIME?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the holdoff time of the Nth edge trigger to 15 ns.

Command message:

*:TRIGger:NEDGe:HLDTIME 1.50E-08*  
*TRIG:NEDG:HLDT 1.50E-08*

Query message:

*TRIG:NEDG:HLDT?*

Response message:

*1.50E-08*

**RELATED COMMANDS**

:TRIGger:NEDGe:HOLDoff

**:TRIGger:NEDGe:HLDEvent****Command/Query****DESCRIPTION**

This command sets the number of holdoff events of the Nth edge trigger.

The query returns the current number of holdoff events of the Nth edge trigger.

**COMMAND SYNTAX**

:TRIGger:NEDGe:HLDEvent <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 100000000].

**QUERY SYNTAX**

:TRIGger:NEDGe:HLDEvent?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the number of holdoff events of the Nth edge trigger to 3.

Command message:

```
:TRIGger:NEDGe:HLDEvent 3  
TRIG:NEDG:HLDEV 3
```

Query message:

```
TRIG:NEDG:HLDEV?
```

Response message:

```
3
```

**RELATED COMMANDS**

:TRIGger:NEDGe:HOLDoff

**:TRIGger:NEDGe:HSTart****Command/Query****DESCRIPTION**

The command defines the initial position of the Nth edge trigger holdoff.

The query returns the initial position of the Nth edge trigger holdoff.

**COMMAND SYNTAX**

:TRIGger:NEDGe:HSTart <start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

- ◆ LAST\_TRIG means the initial position of holdoff is the time of the last trigger.
- ◆ ACQ\_START means the initial position of holdoff is the first time point satisfying the trigger condition.

**QUERY SYNTAX**

:TRIGger:NEDGe:HSTart?

**RESPONSE FORMAT**

<start\_holdoff>

<start\_holdoff>:= {LAST\_TRIG|ACQ\_START}

**EXAMPLE**

The following command sets the start holdoff mode to last trigger.

Command message:

*:TRIGger:NEDGe:HSTart LAST\_TRIG  
TRIG:NEDG:HST LAST\_TRIG*

Query message:

*TRIG:NEDG:HST?*

Response message:

*LAST\_TRIG*

**RELATED COMMANDS**

:TRIGger:NEDGe:HOLDoff

**:TRIGger:NEDGe:NREJect****Command/Query****DESCRIPTION**

The command sets the state of the noise rejection.

The query returns the current state of the noise rejection.

**COMMAND SYNTAX**

:TRIGger:NEDGe:NREJect <state>

<state>:= {OFF|ON}

**QUERY SYNTAX**

:TRIGger:NEDGe:NREJect?

**RESPONSE FORMAT**

<state>

<state>:= {OFF|ON}

**EXAMPLE**

The following command turns on noise rejection.

Command message:

*:TRIGger:NEDGe:NREJect ON*  
*TRIG:NEDG:NREJ ON*

Query message:

*TRIG:NEDG:NREJ?*

Response message:

*ON*

## **:TRIGger:SHOLd Commands**

The :TRIGGER:SHOLd subsystem commands control the setup/hold trigger parameters.

- ◆ **:TRIGger:SHOLd:TYPE**
- ◆ **:TRIGger:SHOLd:CSource**
- ◆ **:TRIGger:SHOLd:CTHReshold**
- ◆ **:TRIGger:SHOLd:SLOPe**
- ◆ **:TRIGger:SHOLd:DSource**
- ◆ **:TRIGger:SHOLd:DTHReshold**
- ◆ **:TRIGger:SHOLd:LEVel**
- ◆ **:TRIGger:SHOLd:LIMit**
- ◆ **:TRIGger:SHOLd:TUPPer**
- ◆ **:TRIGger:SHOLd:TLOWer**

**:TRIGger:SHOLd:TYPE****Command/Query****DESCRIPTION**

The command sets the trigger type of the setup/hold trigger.

The query returns the current the trigger type of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:TYPE <type>

<type>:= {SETup|HOLD}

**QUERY SYNTAX**

:TRIGger:SHOLd:TYPE?

**RESPONSE FORMAT**

<slope>

<slope>:= {SETup|HOLD}

**EXAMPLE**

The following command sets the setup type of the setup/hold trigger.

Command message:

```
:TRIGger:SHOLd:TYPE SETup  
TRIG:SHOL:TYPE SET
```

Query message:

```
TRIG:SHOL:TYPE?
```

Response message:

```
SETup
```

**:TRIGger:SHOLd:CSource****Command/Query****DESCRIPTION**

The command sets the clock source of the setup/hold trigger.

The query returns the current clock source of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:CSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SHOLd:CSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the clock source of the setup/hold trigger as C1.

Command message:

*:TRIGger:SHOLd:CSource C1*  
*TRIG:SHOL:CS C1*

Query message:

*TRIG:SHOL:CS?*

Response message:

*C1*

**RELATED COMMANDS**

:TRIGger:SHOLd:CTHReshold

**:TRIGger:SHOLd:CTHReshold****Command/Query****DESCRIPTION**

The command sets the threshold of clock source of the setup/hold trigger.

The query returns the current threshold of clock source of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:CTHReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                 | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SHOLd:CTHReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of clock source of setup/hold trigger to 1.5 V.

Command message:

```
:TRIGger:SHOLd:CTHReshold 1.50E+00
TRIG:SHOL:CTHR 1.50E+00
```

Query message:

```
TRIG:SHOL:CTHR?
```

Response message:

```
1.50E+00
```

**RELATED COMMANDS**

:TRIGger:SHOLd:CSource



**:TRIGger:SHOLd:SLOPe**

**Command/Query**

**DESCRIPTION**

The command sets the clock slope of the setup/hold trigger.

The query returns the current the clock slope of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:SLOPe <slope\_type>

<slope\_type>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:SHOLd:SLOPe?

**RESPONSE FORMAT**

<slope\_type>

<slope\_type>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the clock slope of the setup/hold trigger to rising.

Command message:

*:TRIGger:SHOLd:SLOPe RISing*  
*TRIG:SHOL:SLOP RIS*

Query message:

*TRIG:SHOL:SLOP?*

Response message:

*RISing*

**:TRIGger:SHOLd:DSource**

**Command/Query**

**DESCRIPTION**

The command sets the data source of the setup/hold trigger.

The query returns the current data source of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:DSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SHOLd:DSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the data source of the setup/hold trigger as C1.

Command message:

*:TRIGger:SHOLd:DSource C1*  
*TRIG:SHOL:DS C1*

Query message:

*TRIG:SHOL:DS?*

Response message:

*C1*

**RELATED COMMANDS**

:TRIGger:SHOLd:DTHReshold

**:TRIGger:SHOLd:DTHReshold****Command/Query****DESCRIPTION**

The command sets the threshold of data source of the setup/hold trigger.

The query returns the current threshold of data source of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:DTHReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                                 | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SHOLd:DTHReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of data source of setup/hold trigger to 1.5 V.

Command message:

```
:TRIGger:SHOLd:DTHReshold 1.50E+00
TRIG:SHOL:DTHR 1.50E+00
```

Query message:

```
TRIG:SHOL:DTHR?
```

Response message:

```
1.50E+00
```

**RELATED COMMANDS**

:TRIGger:SHOLd:DSource

**:TRIGger:SHOLd:LEVel****Command/Query****DESCRIPTION**

The command sets the level state of data source of the setup/hold trigger.

The query returns the current level state of data source of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:LEVel <level\_state>

<level\_value>:= {LOW|HIGH}

**QUERY SYNTAX**

:TRIGger:SHOLd:LEVel?

**RESPONSE FORMAT**

<level\_state>

<level\_value>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the high level of data source of the setup/hold trigger.

Command message:

*:TRIGger:SHOLd:LEVel HIGH*

*TRIG:SHOL:LEV HIGH*

Query message:

*TRIG:SHOL:LEV?*

Response message:

*HIGH*

**:TRIGger:SHOLd:LIMit****Command/Query****DESCRIPTION**

The command sets the limit range type of the setup/hold trigger.

The query returns the current limit range type of the setup/hold trigger.

**COMMAND SYNTAX**

:TRIGger:SHOLd:LIMit <type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**QUERY SYNTAX**

:TRIGger:SHOLd:LIMit?

**RESPONSE FORMAT**

<type>

<type>:= {LESSthan|GREATERthan|INNER|OUTer}

**EXAMPLE**

The following command sets the limit of the setup/hold trigger to LESSthan.

Command message:

*:TRIGger:SHOLd:LIMit LESSthan*

*TRIG:SHOL:LIM LESS*

Query message:

*TRIG:SHOL:LIM?*

Response message:

*LESSthan*

**RELATED COMMANDS**

:TRIGger:SHOLd:TLOWer

:TRIGger:SHOLd:TUPPer

**:TRIGger:SHOLd:TUPPer****Command/Query****DESCRIPTION**

The command sets the upper value of the setup/hold trigger limit type.

The query returns the current upper value of the setup/hold trigger limit type.

**COMMAND SYNTAX**

:TRIGger:SHOLd:TUPPer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [2.00E-09, 2.00E+01].

**Note:**

- The upper value cannot be less than the lower value using by the command :TRIGger:SHOLd:TLOWer.
- The command is not valid when the limit range type is GREATerthan.

**QUERY SYNTAX**

:TRIGger:SHOLd:TUPPer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the upper value of the setup/hold trigger to 30 ns, when the limit range type is OUTer.

Command message:

```
:TRIGger:SHOLd:TUPPer 3.00E-08
TRIG:SHOL:TUPP 3.00E-08
```

Query message:

```
TRIG:SHOL:TUPP?
```

Response message:

```
3.00E-08
```

**RELATED COMMANDS**

```
:TRIGger:SHOLd:LIMit
:TRIGger:SHOLd:TLOWer
```

**:TRIGger:SHOLd:TLOWer****Command/Query****DESCRIPTION**

The command sets the lower value of the setup/hold trigger limit type.

The query returns the current lower value of the setup/hold trigger limit type.

**COMMAND SYNTAX**

:TRIGger:SHOLd:TLOWer <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2. The range of the value is [2.00E-09, 2.00E+01].

**Note:**

- The lower value cannot be greater than the upper value using by the command :TRIGger:SHOLd:TUPPer.
- The command is not valid when the limit range type is LESSthan.

**QUERY SYNTAX**

:TRIGger:SHOLd:TLOWer?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the lower time of the setup/hold trigger to 10 ns.

Command message:

*:TRIGger:SHOLd:TLOWer 1.00E-08*

*TRIG:SHOL:TLOW 1.00E-08*

Query message:

*TRIG:SHOL:TLOW?*

Response message:

*1.00E-08*

**RELATED COMMANDS**

:TRIGger:SHOLd:LIMit

:TRIGger:SHOLd:TUPPer

**:TRIGger:IIC Commands**

The :TRIGGER:IIC subsystem commands control the IIC bus trigger parameters.

- ◆ **:TRIGger:IIC:ADDRess**
- ◆ **:TRIGger:IIC:ALENgtH**
- ◆ **:TRIGger:IIC:CONDition**
- ◆ **:TRIGger:IIC:DAT2**
- ◆ **:TRIGger:IIC:DATA**
- ◆ **:TRIGger:IIC:DLENgtH**
- ◆ **:TRIGger:IIC:LIMit**
- ◆ **:TRIGger:IIC:RWBit**
- ◆ **:TRIGger:IIC:SCLSource**
- ◆ **:TRIGger:IIC:SCLThreshold**
- ◆ **:TRIGger:IIC:SDASource**
- ◆ **:TRIGger:IIC:SDAThreshoLd**



**:TRIGger:IIC:ADDRess****Command/Query****DESCRIPTION**

The command sets the address of the IIC bus trigger.

The query returns the current address of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:ADDRess <addr>

<addr>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 1024], where 1024 represents 0xXX.

**QUERY SYNTAX**

:TRIGger:IIC:ADDRess?

**RESPONSE FORMAT**

<addr>

<addr>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the address of the IIC bus trigger to 0x0a.

Command message:

```
:TRIGger:IIC:ADDRess 10  
TRIG:IIC:ADDR 10
```

Query message:

```
TRIG:IIC:ADDR?
```

Response message:

```
10
```

**RELATED COMMANDS**

:TRIGger:IIC:CONDition

**:TRIGger:IIC:ALENgth****Command/Query****DESCRIPTION**

The command sets the length of address of the IIC bus trigger.

The query returns the current length of address of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:ALENgth <length>

<length>:= {7BIT|10BIT}

**QUERY SYNTAX**

:TRIGger:IIC:ALENgth?

**RESPONSE FORMAT**

<addr\_length>

<addr\_length>:= {7BIT|10BIT}

**EXAMPLE**

The following command sets the length of address of the IIC bus trigger to 10 bit.

Command message:

*:TRIGger:IIC:ALENgth 10BIT*  
*TRIG:IIC:ALEN 10BIT*

Query message:

*TRIG:IIC:ALEN?*

Response message:

*10BIT*

**RELATED COMMANDS**

:TRIGger:IIC:CONDition

**:TRIGger:IIC:CONDition**

**Command/Query**

**DESCRIPTION**

The command sets the trigger condition of the IIC bus.

The query returns the current trigger condition of the IIC bus.

**COMMAND SYNTAX**

:TRIGger:IIC:CONDition <condition>

<condition>:=

{START|STOP|REStart|NACK|EEPROM|7ADDRESS|10ADDRESS|DLENGTH}

**QUERY SYNTAX**

:TRIGger:IIC:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:=

{START|STOP|REStart|NACK|EEPROM|7ADDRESS|10ADDRESS|DLENGTH}

**EXAMPLE**

The following command sets the condition of the IIC bus trigger to STOP.

Command message:

*:TRIGger:IIC:CONDition STOP*

*TRIG:IIC:COND STOP*

Query message:

*TRIG:IIC:COND?*

Response message:

*STOP*

**:TRIGger:IIC:DAT2****Command/Query****DESCRIPTION**

The command sets the data2 of the IIC bus trigger.

The query returns the current data2 of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:DAT2 <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data2 value.

**QUERY SYNTAX**

:TRIGger:IIC:DAT2?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data2 of the IIC bus trigger to 0x0b.

Command message:  
*:TRIGger:IIC:DAT2 11*  
*TRIG:IIC:DAT2 11*

Query message:  
*TRIG:IIC:DAT2?*

Response message:  
*11*

**RELATED COMMANDS**

:TRIGger:IIC:CONDition

**:TRIGger:IIC:DATA****Command/Query****DESCRIPTION**

The command sets the data of the IIC bus trigger.

The query returns the current data of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:DATA <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:IIC:DATA?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data of the IIC bus trigger to 0x2A.

Command message:

*:TRIGger:IIC:DATA 42*  
*TRIG:IIC:DATA 42*

Query message:

*TRIG:IIC:DATA?*

Response message:

*42*

**RELATED COMMANDS**

:TRIGger:IIC:CONDition

:TRIGger:IIC:DAT2

**:TRIGger:IIC:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the IIC bus trigger.

The query returns the current data length of the IIC bus trigger.

**COMMAND SYNTAX**

**:TRIGger:IIC:DLENgth** <length>

<length>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 12].

**QUERY SYNTAX**

**:TRIGger:IIC:DLENgth?**

**RESPONSE FORMAT**

<length>

<length>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the IIC bus trigger to 10 bytes.

Command message:

*:TRIGger:IIC:DLENgth 10*

*TRIG:IIC:DLEN 10*

Query message:

*TRIG:IIC:DLEN?*

Response message:

*10*

**RELATED COMMANDS**

**:TRIGger:IIC:CONDition**

**:TRIGger:IIC:LIMit****Command/Query****DESCRIPTION**

The command sets the data comparison type when the trigger condition is EEPROM on the IIC bus trigger.

The query returns the current the limit range type when the trigger condition is EEPROM.

**COMMAND SYNTAX**

:TRIGger:IIC:LIMit <limit\_type>

<limit\_type>:= {EQUal|GREaterthan|LESSthan}

**QUERY SYNTAX**

:TRIGger:IIC:LIMit?

**RESPONSE FORMAT**

<limit\_type>

<limit\_type>:= {EQUal|GREaterthan|LESSthan}

**EXAMPLE**

The following command sets the limit range type when the trigger condition is EEPROM to LESSthan.

Command message:

*:TRIGger:IIC:LIMit LESSthan*

*TRIG:IIC:LIM LESS*

Query message:

*TRIG:IIC:LIM?*

Response message:

*LESSthan*

**RELATED COMMANDS**

:TRIGger:IIC:CONDition

**:TRIGger:IIC:RWBit****Command/Query****DESCRIPTION**

The command sets whether the trigger frame is read address or write address when the IIC trigger condition is 7 or 10 ADDR&DATA.

The query returns the current read write bit of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:RWBit <type>

<type>:= {WRITe|READ|ANY}

**QUERY SYNTAX**

:TRIGger:IIC:RWBit?

**RESPONSE FORMAT**

<type>

<type>:= {WRITe|READ|ANY}

**EXAMPLE**

The following command sets to trigger on the read address of the IIC bus.

Command message:

*:TRIGger:IIC:RWBit READ  
TRIG:IIC:RWB READ*

Query message:

*TRIG:IIC:RWB?*

Response message:

*READ*

**RELATED COMMANDS**

:TRIGger:IIC:CONDition



**:TRIGger:IIC:SCLSource****Command/Query****DESCRIPTION**

The command selects the SCL source of the IIC bus trigger.

This query returns the current SCL source of the IIC bus trigger.

**COMMAND SYNTAX**

**:TRIGger:IIC:SCLSource <source>**

**<source>:= {C<n>|D<d>}**

**<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.**

**<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.**

**QUERY SYNTAX**

**:TRIGger:IIC:SCLSource?**

**RESPONSE FORMAT**

**<source>**

**<source>:= {C<n>|D<d>}**

**EXAMPLE**

The following command selects the SCL source of the IIC bus trigger as channel 2.

Command message:

*:TRIGger:IIC:SCLSource C2*

*TRIG:IIC:SCLS C2*

Query message:

*TRIG:IIC:SCLS?*

Response message:

*C2*

**RELATED COMMANDS**

**:TRIGger:IIC:SCLThreshold**

**:TRIGger:IIC:SDASource**

**:TRIGger:IIc:SCLThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the SCL on IIC bus trigger.  
 This query returns the current threshold of the SCL on IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIc:SCLThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:IIc:SCLThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the SCL on IIC bus trigger to 1.5 V.

Command message:

*:TRIGger:IIc:SCLThreshold 1.50E+00*  
*TRIG:IIc:SCLT 1.50E+00*

Query message:

*TRIG:IIc:SCLT?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:IIc:SCLSource

**:TRIGger:IIC:SDASource****Command/Query****DESCRIPTION**

The command selects the SDA source of the IIC bus trigger.

This query returns the current SDA source of the IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:SDASource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:IIC:SDASource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the SDA source of the IIC bus trigger as channel 2.

Command message:

*:TRIGger:IIC:SDASource C2*  
*TRIG:IIC:SDAS C2*

Query message:

*TRIG:IIC:SDAS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:IIC:SCLSource

:TRIGger:IIC:SDAThreshold

**:TRIGger:IIC:SDAThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the SDA on IIC bus trigger.  
This query returns the current threshold of the SDA on IIC bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIC:SDAThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:IIC:SDAThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the SDA on IIC bus trigger to 1.5 V.

Command message:

*:TRIGger:IIC:SDAThreshold 1.50E+00*  
*TRIG:IIC:SDAT 1.50E+00*

Query message:

*TRIG:IIC:SDAT?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:IIC:SDASource

## **:TRIGger:SPI Commands**

The :TRIGGER:SPI subsystem commands control the SPI bus trigger modes and parameters.

- ◆ **:TRIGger:SPI:BITOrder**
- ◆ **:TRIGger:SPI:CLKSource**
- ◆ **:TRIGger:SPI:CLKThreshold**
- ◆ **:TRIGger:SPI:CSSource**
- ◆ **:TRIGger:SPI:CSThreshold**
- ◆ **:TRIGger:SPI:CSType**
- ◆ **:TRIGger:SPI:DATA**
- ◆ **:TRIGger:SPI:DLENgth**
- ◆ **:TRIGger:SPI:LATChedge**
- ◆ **:TRIGger:SPI:MISOSource**
- ◆ **:TRIGger:SPI:MISOThreshold**
- ◆ **:TRIGger:SPI:MOSISource**
- ◆ **:TRIGger:SPI:MOSIThreshold**
- ◆ **:TRIGger:SPI:NCSSource**
- ◆ **:TRIGger:SPI:NCSThreshold**
- ◆ **:TRIGger:SPI:TTPe**

**:TRIGger:SPI:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order of the SPI bus trigger.

The query returns the current bit order of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:BITorder <bit\_order>

<bit\_order>:= {LSM|MSB}

**QUERY SYNTAX**

:TRIGger:SPI:BITorder?

**RESPONSE FORMAT**

<bit\_order>

<bit\_order>:= {LSM|MSB}

**EXAMPLE**

The following command sets the bit order of the SPI bus trigger to LSB.

Command message:

*:TRIGger:SPI:BITorder LSB*

*TRIG:SPI:BIT LSB*

Query message:

*TRIG:SPI:BIT?*

Response message:

*LSB*

**:TRIGger:SPI:CLKSource****Command/Query****DESCRIPTION**

The command selects the CLK source of the SPI bus trigger.

This query returns the current CLK source of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CLKSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SPI:CLKSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the CLK source of the SPI bus trigger as channel 2.

Command message:

```
:TRIGger:SPI:CLKSource C2  
TRIG:SPI:CLKS C2
```

Query message:

```
TRIG:SPI:CLKS?
```

Response message:

```
C2
```

**RELATED COMMANDS**

:TRIGger:SPI:CLKThreshold

**:TRIGger:SPI:CLKThreshold****Command/Query****DESCRIPTION**

The command sets the threshold of the CLK on SPI bus trigger.

This query returns the current threshold of the CLK on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CLKThreshold <clk\_threshold>

<clk\_threshold>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:SPI:CLKThreshold?

**RESPONSE FORMAT**

<clk\_threshold>

<clk\_threshold>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the CLK on SPI bus trigger to 1.5 V.

Command message:

```
:TRIGger:SPI:CLKThreshold 1.50E+00
TRIG:SPI:CLKT 1.50E+00
```

Query message:

```
TRIG:SPI:CLKT?
```

Response message:

```
1.50E+00
```

**RELATED COMMANDS**

:TRIGger:SPI:CLKSource



**:TRIGger:SPI:CSSource****Command/Query****DESCRIPTION**

The command sets the CS source of the SPI bus trigger.

The query returns the current CS source of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CSSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SPI:CSSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the CS source of the SPI bus trigger as channel 2.

Command message:

*:TRIGger:SPI:CSSource C2*  
*TRIG:SPI:CSS C2*

Query message:

*TRIG:SPI:CSS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:SPI:CSThreshold

**:TRIGger:SPI:CSThreshold****Command/Query****DESCRIPTION**

The command sets the threshold of the CS on SPI bus trigger.

This query returns the current threshold of the CS on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CSThreshold <threshold>

<threshold>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \cdot \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \cdot \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SPI:CSThreshold?

**RESPONSE FORMAT**

<threshold>

<threshold>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the CS on SPI bus trigger to 1.5 V.

Command message:

```
:TRIGger:SPI:CSThreshold 1.50E+00
TRIG:SPI:CST 1.50E+00
```

Query message:

```
TRIG:SPI:CST?
```

Response message:

```
1.50E+00
```

**RELATED COMMANDS**

:TRIGger:SPI:CSSource

**:TRIGger:SPI:CSType****Command/Query****DESCRIPTION**

The command sets the chip selection type of the SPI bus trigger.

This query returns the current chip selection type of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CSType <type>

<type>:= {NCS|CS|TIMEout[,<time>]}

- ◆ CS means set to chip select state
- ◆ NCS means set to non-chip select state
- ◆ TIMEout indicates set to clock timeout status

<time>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value is [1.00E-07, 5.00E-03].

**QUERY SYNTAX**

:TRIGger:SPI:CSType?

**RESPONSE FORMAT**

<type>

<type>:= {NCS|CS|TIMEout[,<time>]}

<time>:= Value in NR3 format.

**EXAMPLE**

The following command sets the chip selection type of the SPI bus trigger to CS.

Command message:

*:TRIGger:SPI:CSType CS*  
*TRIG:SPI:CSTY CS*

Query message:

*TRIG:SPI:CSTY?*

Response message:

*CS*

**:TRIGger:SPI:DATA****Command****DESCRIPTION**

The command sets the data of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:DATA <data>[,<data>[...[,<data>]]]

<data>:= {0|1|X}

**Note:**

- The number of parameters should be consistent with the data length using by the command :TRIGger:SPI:DLEnGth.
- Parameters are assigned to each bit in order from high to low.

**EXAMPLE**

The following command sets the data of the SPI bus trigger to 0x82 when the data length is 8.

Command message:

*:TRIGger:SPI:DATA 1,0,0,0,0,0,1,0*

*TRIG:SPI:DATA 1,0,0,0,0,0,1,0*

**RELATED COMMANDS**

:TRIGger:SPI:DLEnGth

**:TRIGger:SPI:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the SPI bus trigger.

The query returns the current data length of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:DLENgth <data\_length>

<data\_length>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [4, 96].

**QUERY SYNTAX**

:TRIGger:SPI:DLENgth?

**RESPONSE FORMAT**

<data\_length>

<data\_length>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the SPI bus trigger to 10 bit.

Command message:

*:TRIGger:SPI:DLENgth 10*  
*TRIG:SPI:DLEN 10*

Query message:

*TRIG:SPI:DLEN?*

Response message:

*10*

**:TRIGger:SPI:LATChedge****Command/Query****DESCRIPTION**

The command selects the sampling edge of CLK on SPI bus trigger.

This query returns the sampling edge of CLK on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:CLK:LATChedge <slope>

<slope>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:SPI:LATC?

**RESPONSE FORMAT**

<slope>

<slope>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the threshold judgment condition of CLK on SPI bus trigger to RISing.

Command message:

*:TRIGger:SPI:LATChedge RISing*

*:TRIG:SPI:LATC RIS*

Query message:

*:TRIG:SPI:LATC?*

Response message:

*RISing*

**:TRIGger:SPI:MISOSource**  
**Command/Query**

**DESCRIPTION**

The command selects the MISO source of the SPI bus trigger.  
 This query returns the current MISO source of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:MISOSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.  
 <d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SPI:MISOSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the MISO source of the SPI bus trigger as channel 2.

Command message:  
*:TRIGger:SPI:MISOSource C2*  
*TRIG:SPI:MISOS C2*

Query message:  
*TRIG:SPI:MISOS?*

Response message:  
*C2*

**RELATED COMMANDS**

:TRIGger:SPI:MISOThreshold

**:TRIGger:SPI:MISOThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the MISO on SPI bus trigger.

This query returns the current threshold of the MISO on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:MISOThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SPI:MISOThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the MISO on SPI bus trigger to 1.5 V.

Command message:

*:TRIGger:SPI:MISOThreshold 1.50E+00*  
*TRIG:SPI:MISOT 1.50E+00*

Query message:

*TRIG:SPI:MISOT?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:SPI:MISOSource



**:TRIGger:SPI:MOSISource****Command/Query****DESCRIPTION**

The command selects the MOSI source of the SPI bus trigger.

This query returns the current MOSI source of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:MOSISource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SPI:MOSISource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the MOSI source of the SPI bus trigger as channel 2.

Command message:

*:TRIGger:SPI:MOSISource C2*

*TRIG:SPI:MOSIS C2*

Query message:

*TRIG:SPI:MOSIS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:SPI:MOSIThreshold

**:TRIGger:SPI:MOSIThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the MOSI on SPI bus trigger.

The query returns the current threshold of the MOSI on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:MOSIThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SPI:MOSIThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the MOSI on SPI bus trigger to 1.5 V.

Command message:

*:TRIGger:SPI:MOSIThreshold 1.50E+00*  
*TRIG:SPI:MOSIT 1.50E+00*

Query message:

*TRIG:SPI:MOSIT?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:SPI:MOSISource

**:TRIGger:SPI:NCSSource****Command/Query****DESCRIPTION**

The command sets the NCS source of the SPI bus trigger.

The query returns the current NCS source of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:NCSSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SPI:NCSSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the NCS source of the SPI bus trigger as D0.

Command message:

*:TRIGger:SPI:NCSSource D0*

*:TRIG:SPI:NCSS D0*

Query message:

*:TRIG:SPI:NCSS?*

Response message:

*D0*

**RELATED COMMANDS**

:TRIGger:SPI:NCSThreshold

**:TRIGger:SPI:NCSThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the NCS on SPI bus trigger.

This query returns the current threshold of the NCS on SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:NCSThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:SPI:NCSThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the NCS on IIC bus trigger to 1.5 V.

Command message:

*:TRIGger:SPI:NCSThreshold 1.50E+00*  
*TRIG:SPI:NCST 1.50E+00*

Query message:

*TRIG:SPI:NCST?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:SPI:NCSSource

**:TRIGger:SPI:TTYPe****Command/Query****DESCRIPTION**

The command sets the trigger type of the SPI bus trigger.

The query returns the current trigger type of the SPI bus trigger.

**COMMAND SYNTAX**

:TRIGger:SPI:TTYPe <trigger\_type>

<trigger\_type>:= {MISO|MOSI}

**QUERY SYNTAX**

:TRIGger:SPI:TTYPe?

**RESPONSE FORMAT**

<trigger\_type>

<trigger\_type>:= {MISO|MOSI}

**EXAMPLE**

The following command sets the trigger type of the SPI bus trigger to MOSI.

Command message:

```
:TRIGger:SPI:TTYPe MOSI  
TRIG:SPI:TTYP MOSI
```

Query message:

```
TRIG:SPI:TTYP?
```

Response message:

```
MOSI
```

**:TRIGger:UART Commands**

The :TRIGGER:UART subsystem commands control the UART bus trigger parameters.

- ◆ **:TRIGger:UART:BAUD**
- ◆ **:TRIGger:UART:BITOrder**
- ◆ **:TRIGger:UART:CONDition**
- ◆ **:TRIGger:UART:DATA**
- ◆ **:TRIGger:UART:DLENgth**
- ◆ **:TRIGger:UART:IDLE**
- ◆ **:TRIGger:UART:LIMit**
- ◆ **:TRIGger:UART:PARity**
- ◆ **:TRIGger:UART:RXSource**
- ◆ **:TRIGger:UART:**
- ◆ **:TRIGger:UART:STOP**
- ◆ **:TRIGger:UART:TTYPe**
- ◆ **:TRIGger:UART:TXSource**
- ◆ **:TRIGger:UART:**

**:TRIGger:UART:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the UART bus trigger.

The query returns the current baud rate of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:BAUD <baud>

<baud>:=

{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|38400bps|57600bps|115200bps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [300, 20000000].

**QUERY SYNTAX**

:TRIGger:UART:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=

{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|38400bps|57600bps|115200bps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the UART bus trigger to 9600bps.

Command message:

```
:TRIGger:UART:BAUD 9600bps  
TRIG:UART:BAUD 9600bps
```

Query message:

```
TRIG:UART:BAUD?
```

Response message:

```
9600bps
```

**:TRIGger:UART:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order of the UART trigger.

The query returns the current bit order of the UART trigger.

**COMMAND SYNTAX**

:TRIGger:UART:BITorder <order>

<order>:= {LSM|MSB}

**QUERY SYNTAX**

:TRIGger:UART:BITorder?

**RESPONSE FORMAT**

<order>

<order>:= {LSM|MSB}

**EXAMPLE**

The following command sets the bit order to LSB.

Command message:

```
:TRIGger:UART:BITorder LSB  
TRIG:UART:BIT LSB
```

Query message:

```
TRIG:UART:BIT?
```

Response message:

```
LSB
```



**:TRIGger:UART:CONDition****Command/Query****DESCRIPTION**

The command sets the condition of the UART bus trigger.

The query returns the current condition of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:CONDition <condition>

<condition>:= {START|STOP|DATA|ERRor}

**QUERY SYNTAX**

:TRIGger:UART:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {START|STOP|DATA|ERRor}

**EXAMPLE**

The following command sets the condition of the UART bus trigger to STOP.

Command message:

*:TRIGger:UART:CONDition STOP*  
*TRIG:UART:COND STOP*

Query message:

*TRIG:UART:COND?*

Response message:

*STOP*

**:TRIGger:UART:DATA****Command/Query****DESCRIPTION**

The command sets the data of the UART bus trigger.

The query returns the current data of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:DATA <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

- The range of the value is related to data length by using the command :TRIGger:UART:DLENgth.
- Use the don't care data (256, data length is 8) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:UART:DATA?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data of the UART bus trigger to 0x53.

Command message:

```
:TRIGger:UART:DATA 83  
TRIG:UART:DATA 83
```

Query message:

```
TRIG:UART:DATA?
```

Response message:

```
83
```

**RELATED COMMANDS**

:TRIGger:UART:CONDition  
:TRIGger:UART:DLENgth

**:TRIGger:UART:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the UART bus trigger.

The query returns the current data length of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:DLENgth <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [5, 8].

**QUERY SYNTAX**

:TRIGger:UART:DLENgth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the UART bus trigger to 8.

Command message:

*:TRIGger:UART:DLENgth 8*  
*TRIG:UART:DLEN 8*

Query message:

*TRIG:UART:DLEN?*

Response message:

*8*

**:TRIGger:UART:IDLE****Command/Query****DESCRIPTION**

The command sets the idle level of the UART bus trigger.

The query returns the current idle level of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:IDLE <idle>

<idle>:= {LOW|HIGH}

**QUERY SYNTAX**

:TRIGger:UART:IDLE?

**RESPONSE FORMAT**

<idle>

<idle>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the idle level of the UART bus trigger as LOW.

Command message:

*:TRIGger:UART:IDLE LOW*  
*TRIG:UART:IDLE LOW*

Query message:

*TRIG:UART:IDLE?*

Response message:

*LOW*

**:TRIGger:UART:LIMit****Command/Query****DESCRIPTION**

The command sets the data comparison type of the UART bus trigger when the trigger condition is Data.

The query returns the current data comparison type of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:LIMit <limit\_type>

<limit\_type>:= {EQUal|GREaterthan|LESSthan}

**QUERY SYNTAX**

:TRIGger:UART:LIMit?

**RESPONSE FORMAT**

<limit\_type>

<limit\_type>:= {EQUal|GREaterthan|LESSthan}

**EXAMPLE**

The following command sets the limit of the UART bus trigger to LESSthan.

Command message:

*:TRIGger:UART:LIMit LESSthan*  
*TRIG:UART:LIM LESS*

Query message:

*TRIG:UART:LIM?*

Response message:

*LESSthan*

**RELATED COMMANDS**

:TRIGger:UART:CONDition

**:TRIGger:UART:PARity****Command/Query****DESCRIPTION**

The command sets the parity check of the UART bus trigger.

The query returns the current parity check of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:PARity <parity>

<parity>:= {NONE|ODD|EVEN|MARK|SPACE}

**QUERY SYNTAX**

:TRIGger:UART:PARity?

**RESPONSE FORMAT**

<parity\_check>

<parity\_check>:= {NONE|ODD|EVEN|MARK|SPACE}

**EXAMPLE**

The following command sets the parity check of the UART bus trigger to odd.

Command message:

*:TRIGger:UART:PARity ODD*  
*TRIG:UART:PAR ODD*

Query message:

*TRIG:UART:PAR?*

Response message:

*ODD*

**:TRIGger:UART:RXSource****Command/Query****DESCRIPTION**

The command sets the RX source of the UART bus trigger.

The query returns the current RX source of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:RXSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:UART:RXSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command selects the RX source of the UART bus trigger as channel 2.

Command message:

*:TRIGger:UART:RXSource C2*

*TRIG:UART:RXS C2*

Query message:

*TRIG:UART:RXS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:UART:

**:TRIGger:UART:RXTHreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of RX on UART bus trigger.  
The query returns the current threshold of RX on UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:RXTHreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:UART:RXTHreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of RX on UART bus trigger to 1.5 V.

Command message:

*:TRIGger:UART:RXTHreshold 1.50E+00  
TRIG:UART:RXTH 1.50E+00*

Query message:

*TRIG:UART:RXTH?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:UART:RXSource



**:TRIGger:UART:STOP****Command/Query****DESCRIPTION**

The command sets the length of the stop bit on UART bus trigger.

The query returns the current length of the stop bit on UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:STOP <bit>

<bit>:= {1|1.5|2}

**QUERY SYNTAX**

:TRIGger:UART:STOP?

**RESPONSE FORMAT**

<bit>

<bit>:= {1|1.5|2}

**EXAMPLE**

The following command sets the length of the stop bit on UART bus trigger to 1 bit.

Command message:

*:TRIGger:UART:STOP 1*

*TRIG:UART:STOP 1*

Query message:

*TRIG:UART:STOP?*

Response message:

*1*

**:TRIGger:UART:TTYPe****Command/Query****DESCRIPTION**

The command sets the trigger type of the UART bus trigger.

The query returns the current trigger type of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:TTYPe <trigger\_type>

<trigger\_type>:= {RX|TX}

**QUERY SYNTAX**

:TRIGger:UART:TTYPe?

**RESPONSE FORMAT**

<trigger\_type>

<trigger\_type>:= {RX|TX}

**EXAMPLE**

The following command sets the trigger type of the UART bus trigger to RX.

Command message:

```
:TRIGger:UART:TTYPe RX  
TRIG:UART:TTYP RX
```

Query message:

```
TRIG:UART:TTYP?
```

Response message:

```
RX
```

**:TRIGger:UART:TXSource****Command/Query****DESCRIPTION**

The command sets the TX source of the UART bus trigger.

The query returns the current TX source of the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:TXSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:UART:TXSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the TX source of the UART bus trigger as channel 2.

Command message:

*:TRIGger:UART:TXSource C2*

*TRIG:UART:TXS C2*

Query message:

*TRIG:UART:TXS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:UART:

**:TRIGger:UART:TXTHreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of TX on the UART bus trigger.

The query returns the current threshold of TX on the UART bus trigger.

**COMMAND SYNTAX**

:TRIGger:UART:TXTHreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:UART:TXTHreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of TX on UART bus trigger to 1.5 V.

Command message:

*:TRIGger:UART:TXTHreshold 1.50E+00  
TRIG:UART:TXTH 1.50E+00*

Query message:

*TRIG:UART:TXTH?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:UART:TXSource

## **:TRIGger:CAN Commands**

The :TRIGGER:CAN subsystem commands control the CAN bus trigger parameters.

- ◆ **:TRIGger:CAN:BAUD**
- ◆ **:TRIGger:CAN:CONDition**
- ◆ **:TRIGger:CAN:DAT2**
- ◆ **:TRIGger:CAN:DATA**
- ◆ **:TRIGger:CAN:ID**
- ◆ **:TRIGger:CAN:IDLength**
- ◆ **:TRIGger:CAN:SOURce**
- ◆ **:TRIGger:CAN:THReshold**

**:TRIGger:CAN:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the CAN bus trigger.

The command query returns the baud rate of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:BAUD <baud>

<baud>:=

{5kbps|10kbps|20kbps|50kbps|100kbps|125kbps|250kbps|500kbps|800kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [5000, 1000000].

**QUERY SYNTAX**

:TRIGger:CAN:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=

{5kbps|10kbps|20kbps|50kbps|100kbps|125kbps|250kbps|500kbps|800kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

This command sets the baud rate of the CAN bus trigger to 20kbps.

Command message:

```
:TRIGger:CAN:BAUD 20kbps
TRIG:CAN:BAUD 20kbps
```

Query message:

```
TRIG:CAN:BAUD?
```

Response message:

```
20kbps
```

**:TRIGger:CAN:CONDition****Command/Query****DESCRIPTION**

The command sets the trigger condition for the CAN bus trigger.

The query returns the current trigger condition for the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:CONDition <condition>

<condition>:= {START|REMOte|ID|ID\_AND\_DATA|ERRor}

**QUERY SYNTAX**

:TRIGger:CAN:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {START|REMOte|ID|ID\_AND\_DATA|ERRor}

**EXAMPLE**

The following command sets the trigger condition for the CAN bus trigger to start.

Command message:

*:TRIGger:CAN:CONDition START*

*TRIG:CAN:COND STAR*

Query message:

*TRIG:CAN:COND?*

Response message:

*START*

**:TRIGger:CAN:DAT2****Command/Query****DESCRIPTION**

The command sets the data2 of the CAN bus trigger.

The query returns the current data2 of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:DAT2 <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data2 value.

**QUERY SYNTAX**

:TRIGger:CAN:DAT2?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the CAN bus triggered data 2 to 0x49.

Command message:

*:TRIGger:CAN:DAT2 73*

*TRIG:CAN:DAT2 73*

Query message:

*TRIG:CAN:DAT2?*

Response message:

*73*

**RELATED COMMANDS**

:TRIGger:CAN:CONDition



**:TRIGger:CAN:DATA****Command/Query****DESCRIPTION**

The command sets the data of the CAN bus trigger.

The query returns the current data of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:DATA <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:CAN:DATA?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data of the CAN bus triggered to 0x43.

Command message:

*:TRIGger:CAN:DATA 67*  
*TRIG:CAN:DATA 67*

Query message:

*TRIG:CAN:DATA?*

Response message:

*67*

**RELATED COMMANDS**

:TRIGger:CAN:CONDition

**:TRIGger:CAN:ID****Command/Query****DESCRIPTION**

The command sets the ID of the CAN bus trigger.

The query returns the current ID of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:ID <id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

When the ID length is 29 bits, the range of the value is [0, 536870912] where 536870912 represents 0XXXXXXXXX. When the ID length is 11 bits, the range of the value is [0, 2048] where 2048 represents 0XXXXX.

**QUERY SYNTAX**

:TRIGger:CAN:ID?

**RESPONSE FORMAT**

<id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the ID of the CAN bus trigger to 0x7819F51.

Command message:

*:TRIGger:CAN:ID 125935441*

*TRIG:CAN:ID 125935441*

Query message:

*TRIG:CAN:ID?*

Response message:

*125935441*

**RELATED COMMANDS**

:TRIGger:CAN:CONDition

**:TRIGger:CAN:IDLength****Command/Query****DESCRIPTION**

The command sets the ID length of the CAN bus trigger when the trigger condition is Remote, ID or ID+Data.

The query returns the current ID length of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:IDLENgth <id\_length>

<id\_length>:= {11BITS|29BITS}

**QUERY SYNTAX**

:TRIGger:CAN:IDLENgth?

**RESPONSE FORMAT**

<id\_length>

<id\_length>:= {11BITS|29BITS}

**EXAMPLE**

The following command sets the ID length of the CAN trigger to 29BITS.

Command message:

```
:TRIGger:CAN:IDLength 29BITS  
TRIG:CAN:IDL 29BITS
```

Query message:

```
TRIG:CAN:IDL?
```

Response message:

```
29BITS
```

**RELATED COMMANDS**

:TRIGger:CAN:CONDition

**:TRIGger:CAN:SOURce**

**Command/Query**

**DESCRIPTION**

The command selects the source of the CAN bus trigger.

The query returns the current source of the CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:CAN:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

This following command sets the source of the CAN bus trigger to C2.

Command message:

*:TRIGger:CAN:SOURce C2*  
*TRIG:CAN:SOUR C2*

Query message:

*TRIG:CAN:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:CAN:THReshold

**:TRIGger:CAN:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on CAN bus trigger.

The query returns the current threshold of the source on CAN bus trigger.

**COMMAND SYNTAX**

:TRIGger:CAN:THReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:CAN:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

**EXAMPLE**

The following command sets the threshold of the source on CAN bus trigger to 1.5 V.

Command message:

*:TRIGger:CAN:THReshold 1.50E+00*  
*TRIG:CAN:THR 1.50E+00*

Query message:

*TRIG:CAN:THR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:CAN:SOURce

**:TRIGger:LIN Commands**

The :TRIGGER:LIN subsystem commands control the LIN bus trigger parameters.

- ◆ **:TRIGger:LIN:BAUD**
- ◆ **:TRIGger:LIN:CONDition**
- ◆ **:TRIGger:LIN:DAT2**
- ◆ **:TRIGger:LIN:DATA**
- ◆ **:TRIGger:LIN:ERRor:CHECksum**
- ◆ **:TRIGger:LIN:ERRor:DLENgth**
- ◆ **:TRIGger:LIN:ERRor:ID**
- ◆ **:TRIGger:LIN:ERRor:PARity**
- ◆ **:TRIGger:LIN:ERRor:SYNC**
- ◆ **:TRIGger:LIN:ID**
- ◆ **:TRIGger:LIN:SOURce**
- ◆ **:TRIGger:LIN:STANdard**
- ◆ **:TRIGger:LIN:THReshold**

**:TRIGger:LIN:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the LIN bus trigger.

The query returns the current baud rate of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:BAUD <baud>

<baud>:=

{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|  
CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [300, 20000000].

**QUERY SYNTAX**

:TRIGger:LIN:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:=

{600bps|1200bps|2400bps|4800bps|9600bps|19200bps|  
CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the LIN bus trigger to 9600bps.

Command message:

*:TRIGger:LIN:BAUD 9600bps*  
*:TRIG:LIN:BAUD 9600bps*

Query message:

*:TRIG:LIN:BAUD?*

Response message:

*9600bps*

**:TRIGger:LIN:CONDition****Command/Query****DESCRIPTION**

The command sets the trigger condition of the LIN bus.

The query returns the current trigger condition of the LIN bus.

**COMMAND SYNTAX**

:TRIGger:LIN:CONDition <condition>

<condition>:= {BReak|ID|ID\_AND\_DATA|DATA\_ERROR}

**QUERY SYNTAX**

:TRIGger:LIN:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {BReak|ID|ID\_AND\_DATA|DATA\_ERROR}

**EXAMPLE**

The following command sets the condition of the LIN bus trigger to ID\_AND\_DATA.

Command message:

```
:TRIGger:LIN:CONDition ID_AND_DATA  
TRIG:LIN:COND ID_AND_DATA
```

Query message:

```
TRIG:LIN:COND?
```

Response message:

```
ID_AND_DATA
```



**:TRIGger:LIN:DAT2****Command/Query****DESCRIPTION**

The command sets the data2 of the LIN bus trigger when the trigger condition is ID+Data.

The query returns the current data2 of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:DAT2 <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data2 value.

**QUERY SYNTAX**

:TRIGger:LIN:DAT2?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data2 of the LIN bus trigger to 0x4C.

Command message:

*:TRIGger:LIN:DAT2 76*

*TRIG:LIN:DAT2 76*

Query message:

*TRIG:LIN:DAT2?*

Response message:

*76*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition

:TRIGger:LIN:DATA

**:TRIGger:LIN:DATA****Command/Query****DESCRIPTION**

The command sets the data of the LIN bus trigger when the trigger condition is ID+Data.

The query returns the current data1 of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:DATA <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:LIN:DATA?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data1 of the LIN bus trigger to 0x45.

Command message:

*:TRIGger:LIN:DATA 69*

*TRIG:LIN:DATA 69*

Query message:

*TRIG:LIN:DATA?*

Response message:

*69*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition

:TRIGger:LIN:DAT2

**:TRIGger:LIN:ERRor:CHECksum**

**Command/Query**

**DESCRIPTION**

The command sets the checksum error state of the LIN bus trigger when the trigger condition is Error.

The query returns the current checksum error state of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:ERRor:CHECksum <state>

<state>:= {0|1}

- ◆ 0 means OFF
- ◆ 1 means ON

**QUERY SYNTAX**

:TRIGger:LIN:ERRor:CHECksum?

**RESPONSE FORMAT**

<state>

<state>:= {0|1}

**EXAMPLE**

The following command sets to trigger when a checksum error occurs.

Command message:

*:TRIGger:LIN:ERRor:CHECksum 1*  
*TRIG:LIN:ERR:CHEC 1*

Query message:

*TRIG:LIN:ERR:CHEC?*

Response message:

*1*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition

**:TRIGger:LIN:ERRor:DLENgth****Command/Query****DESCRIPTION**

The command sets the data length of the error frame when the trigger condition is Error and the checksum error state is on.

The query returns the current data length of the error frame on LIN bus.

**COMMAND SYNTAX**

:TRIGger:LIN:ERRor:DLENgth <length>

<length>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1, 8].

**QUERY SYNTAX**

:TRIGger:LIN:ERRor:DLENgth?

**RESPONSE FORMAT**

<length>

<length>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data length of the error frame on LIN bus to 4 bytes.

Command message:

*:TRIGger:LIN:ERRor:DLENgth 4*  
*TRIG:LIN:ERR:DLEN 4*

Query message:

*TRIG:LIN:ERR:DLEN?*

Response message:

*4*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition  
:TRIGger:LIN:ERRor:CHECksum

**:TRIGger:LIN:ERRor:ID****Command/Query****DESCRIPTION**

The command sets the error frame ID of the LIN bus when the trigger condition is Error and the checksum error state is on.

The query returns the current error frame ID of the LIN bus.

**COMMAND SYNTAX**

:TRIGger:LIN:ERRor:ID <id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 63].

**QUERY SYNTAX**

:TRIGger:LIN:ERRor:ID?

**RESPONSE FORMAT**

<id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the error frame ID of the LIN bus trigger to 0x2A.

Command message:

*:TRIGger:LIN:ERRor:ID 42*  
*TRIG:LIN:ERR:ID 42*

Query message:

*TRIG:LIN:ERR:ID?*

Response message:

*42*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition  
:TRIGger:LIN:ERRor:CHECksum

**:TRIGger:LIN:ERRor:PARity****Command/Query****DESCRIPTION**

The command sets the header parity error state of the LIN bus trigger when the trigger condition is Error.

The query returns the header parity error state of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:ERRor:PARity <state>

<state>:= {0|1}

- ◆ 0 means OFF
- ◆ 1 means ON

**QUERY SYNTAX**

:TRIGger:LIN:ERRor:PARity?

**RESPONSE FORMAT**

<state>

<state>:= {0|1}

**EXAMPLE**

The following command sets to trigger when a header parity error occurs.

Command message:

*:TRIGger:LIN:ERRor:PARity 1*

*TRIG:LIN:ERR:PAR 1*

Query message:

*TRIG:LIN:ERR:PAR?*

Response message:

*1*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition

**:TRIGger:LIN:ERRor:SYNC****Command/Query****DESCRIPTION**

The command sets the sync byte error state of the LIN bus trigger.

The query returns the current sync byte error state of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:ERRor:SYNC <state>

<state>:= {0|1}

**QUERY SYNTAX**

:TRIGger:LIN:ERRor:SYNC?

**RESPONSE FORMAT**

<state>

<state>:= {0|1}

**EXAMPLE**

The following command sets to trigger when a sync byte error occurs.

Command message:

*:TRIGger:LIN:ERRor:SYNC 1*

*TRIG:LIN:ERR:SYNC 1*

Query message:

*TRIG:LIN:ERR:SYNC?*

Response message:

*1*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition

**:TRIGger:LIN:ID****Command/Query****DESCRIPTION**

The command sets the ID of the LIN bus when the trigger condition is ID.

The query returns the current ID of the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:ID <id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 64].

**Note:**

Use the don't care data (64) to ignore the ID value.

**QUERY SYNTAX**

:TRIGger:LIN:ID?

**RESPONSE FORMAT**

<id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the ID of the LIN bus trigger to 0x2B.

Command message:

*:TRIGger:LIN:ID 43*

*TRIG:LIN:ID 43*

Query message:

*TRIG:LIN:ID?*

Response message:

*43*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition



**:TRIGger:LIN:SOURce****Command/Query****DESCRIPTION**

The command selects the trigger source of the LIN bus.

The query returns the current trigger source of the LIN bus.

**COMMAND SYNTAX**

:TRIGger:LIN:Source <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:LIN:Source?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the trigger source of the LIN bus as channel 2.

Command message:

*:TRIGger:LIN:SOURce C2*  
*TRIG:LIN:SOUR C2*

Query message:

*TRIG:LIN:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:LIN:THReshold

**:TRIGger:LIN:STANdard****Command/Query****DESCRIPTION**

The command sets the LIN protocol standard when the trigger condition is Error and the checksum error state is on.

The query returns the current protocol standard of the LIN bus.

**COMMAND SYNTAX**

:TRIGger:LIN:STANdard <version>

<version>:= {0|1}

- ◆ 0 means Rev1.3
- ◆ 1 means Rev2.x

**QUERY SYNTAX**

:TRIGger:LIN:STANdard?

**RESPONSE FORMAT**

<version>

<version>:= {0|1}

**EXAMPLE**

The following command sets to trigger when a checksum error occurs according to Lin protocol 1.3.

Command message:

*:TRIGger:LIN:STANdard 0*  
*TRIG:LIN:STAN 0*

Query message:

*TRIG:LIN:STAN?*

Response message:

*0*

**RELATED COMMANDS**

:TRIGger:LIN:CONDition  
:TRIGger:LIN:ERRor:CHECKsum

**:TRIGger:LIN:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on LIN bus trigger.

The query returns the current threshold of source on the LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:LIN:THReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model  | Value Range   |
|--|---|
| SDS7000A   | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L<br>SHS800X/SHS1000X                              | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD<br>SDS1000X HD<br>SDS800X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:LIN:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the source on LIN bus trigger to 1.5 V.

Command message:

*:TRIGger:LIN:THReshold 1.50E+00*  
*TRIG:LIN:THR 1.50E+00*

Query message:

*TRIG:LIN:THR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:LIN:SOURce

**:TRIGger:FLEXray Commands [Option]**

The :TRIGGER:FLEXray subsystem commands control the FlexRay bus trigger parameters.

- ◆ **:TRIGger:FLEXray:BAUD**
- ◆ **:TRIGger:FLEXray:CONDition**
- ◆ **:TRIGger:FLEXray:FRAMe:COMPare**
- ◆ **:TRIGger:FLEXray:FRAMe:CYCLe**
- ◆ **:TRIGger:FLEXray:FRAMe:ID**
- ◆ **:TRIGger:FLEXray:FRAMe:REPetition**
- ◆ **:TRIGger:FLEXray:SOURce**
- ◆ **:TRIGger:FLEXray:THReshold**

**:TRIGger:FLEXray:BAUD****Command/Query****DESCRIPTION**

The command sets the baud rate of the Flexray bus trigger.

The query returns the current baud rate of the Flexray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:BAUD <baud>

<baud>:= {2500kbps|5Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [1000000, 20000000].

**QUERY SYNTAX**

:TRIGger:FLEXray:BAUD?

**RESPONSE FORMAT**

<baud>

<baud>:= {2500kbps|5Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the baud rate of the Flexray bus trigger to 2500kbps.

Command message:

```
:TRIGger:FLEXray:BAUD 2500kbps  
TRIG:FLEX:BAUD 2500kbps
```

Query message:

```
TRIG:FLEX:BAUD?
```

Response message:

```
2500kbps
```

**:TRIGger:FLEXray:CONDition****Command/Query****DESCRIPTION**

The command sets the trigger condition of FLEXray bus.

The query returns the current trigger condition of FLEXray bus.

**COMMAND SYNTAX**

:TRIGger:FLEXray:CONDition <condition>

<condition>:= {TSS|FRAMe|SYMBol|ERRor}

**QUERY SYNTAX**

:TRIGger:FLEXray:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {TSS|FRAMe|SYMBol|ERRor}

**EXAMPLE**

The following command sets the condition of FLEXray bus trigger to SYMBol.

Command message:

*:TRIGger:FLEXray:CONDition SYMBol*  
*TRIG:FLEX:COND SYMB*

Query message:

*TRIG:FLEX:COND?*

Response message:

*SYMBol*

**:TRIGger:FLEXray:FRAMe:COMPare****Command/Query****DESCRIPTION**

The command sets the frame cycle compare type of FLEXray bus trigger.

The query returns the current frame cycle compare type of FLEXray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:FRAMe:COMPare <type>

<type >:= {ANY|EQUal|GREaterthan|LESSthan}

**QUERY SYNTAX**

:TRIGger:FLEXray:FRAMe:COMPare?

**RESPONSE FORMAT**

<type >

<type >:= {ANY|EQUal|GREaterthan|LESSthan}

**EXAMPLE**

The following command sets the frame cycle compare type of FLEXray bus trigger to LESSthan.

Command message:

*:TRIGger:FLEXray:FRAMe:COMPare LESSthan*  
*TRIG:FLEX:FRAM:COMP LESS*

Query message:

*TRIG:FLEX:FRAM:COMP?*

Response message:

*LESSthan*

**RELATED COMMANDS**

:TRIGger:FLEXray:CONDition

**:TRIGger:FLEXray:FRAMe:CYCLe****Command/Query****DESCRIPTION**

The command sets the frame cycle of FLEXray bus trigger.

The query returns the current frame cycle of FLEXray bus trigger.

**COMMAND SYNTAX**

**:TRIGger:FLEXray:FRAMe:CYCLe <cycle>**

<cycle>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 63].

**QUERY SYNTAX**

**:TRIGger:FLEXray:FRAMe:CYCLe?**

**RESPONSE FORMAT**

<cycle>

<cycle>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the frame cycle of FLEXray bus trigger to 2.

Command message:

```
:TRIGger:FLEXray:FRAMe:CYCLe 2  
TRIG:FLEX:FRAM:CYCL 2
```

Query message:

```
TRIG:FLEX:FRAM:CYCL?
```

Response message:

```
2
```

**RELATED COMMANDS**

**:TRIGger:FLEXray:CONDition**



**:TRIGger:FLEXray:FRAMe:ID****Command/Query****DESCRIPTION**

The command sets the frame ID of FLEXray bus trigger.

The query returns the current frame ID of FLEXray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:FRAMe:ID <id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 2048].

**Note:**

Use the don't care data (2048) to ignore the ID value.

**QUERY SYNTAX**

:TRIGger:FLEXray:FRAMe:ID?

**RESPONSE FORMAT**

<id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the frame ID of FLEXray bus trigger to 0x701.

Command message:

```
:TRIGger:FLEXray:FRAMe:ID 1793  
TRIG:FLEX:FRAM:ID 1793
```

Query message:

```
TRIG:FLEX:FRAM:ID?
```

Response message:

```
1793
```

**RELATED COMMANDS**

:TRIGger:FLEXray:CONDition

**:TRIGger:FLEXray:FRAMe:REPetition****Command/Query****DESCRIPTION**

The command sets the cycle repetition of FLEXray bus trigger when the cycle compare type is Equal

The query returns the current frame repetition of FLEXray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:FRAMe:REPetition <times>

<times>:= {1|2|4|8|16|32|64}

**QUERY SYNTAX**

:TRIGger:FLEXray:FRAMe:REPetition?

**RESPONSE FORMAT**

<times>

<times>:= {1|2|4|8|16|32|64}

**EXAMPLE**

The following command sets the frame repetition of FLEXray bus trigger to 8.

Command message:

```
:TRIGger:FLEXray:FRAMe:REPetition 8
TRIG:FLEX:FRAM:REP 8
```

Query message:

```
TRIG:FLEX:FRAM:REP?
```

Response message:

```
8
```

**RELATED COMMANDS**

:TRIGger:FLEXray:CONDition

:TRIGger:FLEXray:FRAMe:COMPare

**:TRIGger:FLEXray:SOURce****Command/Query****DESCRIPTION**

The command selects the source of FLEXray bus trigger.

The query returns the current source of FLEXray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:Source <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:FLEXray:Source?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the source of FLEXray bus trigger as channel 2.

Command message:

*:TRIGger:FLEXray:SOURce C2*  
*TRIG:FLEX:SOUR C2*

Query message:

*TRIG:FLEX:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:FLEXray:THReshold

**:TRIGger:FLEXray:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on FLEXray bus trigger.

The query returns the current threshold of the source on FLEXray bus trigger.

**COMMAND SYNTAX**

:TRIGger:FLEXray:THReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 * \text{vertical\_scale} - \text{vertical\_offset}, 4.26 * \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 * \text{vertical\_scale} - \text{vertical\_offset}, 4.5 * \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 * \text{vertical\_scale} - \text{vertical\_offset}, 4.1 * \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:FLEXray:THReshold?

**RESPONSE FORMAT**

< value >

< value >:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the source on FLEXray bus trigger to 1.5 V.

Command message:

*:TRIGger:FLEXray:THReshold 1.50E+00*  
*TRIG:FLEX:THR 1.50E+00*

Query message:

*TRIG:FLEX:THR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:FLEXray:SOURce

## **:TRIGger:CANFd Commands [Option]**

The :TRIGGER:CANFd subsystem commands control the CAN FD bus trigger parameters.

- ◆ **:TRIGger:CANFd:BAUDData**
- ◆ **:TRIGger:CANFd:BAUDNominal**
- ◆ **:TRIGger:CANFd:CONDition**
- ◆ **:TRIGger:CANFd:DAT2**
- ◆ **:TRIGger:CANFd:DATA**
- ◆ **:TRIGger:CANFd:FTYPE**
- ◆ **:TRIGger:CANFd:ID**
- ◆ **:TRIGger:CANFd:IDLength**
- ◆ **:TRIGger:CANFd:SOURce**
- ◆ **:TRIGger:CANFd:THReshold**

**:TRIGger:CANFd:BAUDData****Command/Query****DESCRIPTION**

The command sets the data baud rate of the CAN FD bus trigger when the frame type is Both or CAN FD.

The query returns the current data baud rate of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:BAUDData <baud>

<baud>:=  
{500kbps|1Mbps|2Mbps|5Mbps|8Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [100000, 10000000].

**QUERY SYNTAX**

:TRIGger:CANFd:BAUDData?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{500kbps|1Mbps|2Mbps|5Mbps|8Mbps|10Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data baud rate of the CAN FD bus trigger to 500kbps.

Command message:

```
:TRIGger:CANFd:BAUDData 500kbps  
TRIG:CANF:BAUDD 500kbps
```

Query message:

```
TRIG:CANF:BAUDD?
```

Response message:

```
500kbps
```

**RELATED COMMANDS**

```
:TRIGger:CANFd:FTYPE  
:TRIGger:CANFd:BAUDNominal
```

**:TRIGger:CANFd:BAUDNominal****Command/Query****DESCRIPTION**

The command sets the nominal baud rate of the CAN FD bus trigger.

The query returns the current nominal baud rate of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:BAUDNominal <baud>

<baud>:=  
{10kbps|25kbps|50kbps|100kbps|250kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [10000, 1000000].

**QUERY SYNTAX**

:TRIGger:CANFd:BAUDNominal?

**RESPONSE FORMAT**

<baud>

<baud>:=  
{10kbps|25kbps|50kbps|100kbps|250kbps|1Mbps|CUSTom[,<value>]}

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the nominal baud of the CAN FD bus trigger to 10kbps.

Command message:

```
:TRIGger:CANFd:BAUDNominal 10kbps  
TRIG:CANF:BAUDN 10kbps
```

Query message:

```
TRIG:CANF:BAUDN?
```

Response message:

```
10kbps
```

**RELATED COMMANDS**

:TRIGger:CANFd:FTYPE  
:TRIGger:CANFd:BAUDData

**:TRIGger:CANFd:CONDition****Command/Query****DESCRIPTION**

The command sets the trigger condition for the CAN FD bus trigger.

The query returns the current trigger condition for the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:CONDition <condition>

<condition>:= {START|REMOte|ID|ID\_AND\_DATA|ERRor}

**QUERY SYNTAX**

:TRIGger:CANFd:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {START|REMOte|ID|ID\_AND\_DATA|ERRor}

**EXAMPLE**

The following command sets the condition of the CAN FD bus trigger to ID\_AND\_DATA.

Command message:

```
:TRIGger:CANFd:CONDition ID_AND_DATA  
TRIG:CANF:COND ID_AND_DATA
```

Query message:

```
TRIG:CANF:COND?
```

Response message:

```
ID_AND_DATA
```



**:TRIGger:CANFd:DAT2****Command/Query****DESCRIPTION**

The command sets the data2 of the CAN FD bus when the trigger condition is ID+Data.

The query returns the current data2 of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:DAT2 <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data2 value.

**QUERY SYNTAX**

:TRIGger:CANFd:DAT2?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data2 of the CAN FD bus trigger to 0x3F.

Command message:

*TRIGger:CANFd:DAT2 63*

*TRIG:CANF:DAT2 63*

Query message:

*TRIG:CANF:DAT2?*

Response message:

*63*

**RELATED COMMANDS**

:TRIGger:CANFd:CONDition

:TRIGger:CANFd:DATA

**:TRIGger:CANFd:DATA****Command/Query****DESCRIPTION**

The command the data of the CAN FD bus trigger.

The query returns the current data of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:DATA <data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1. The range of the value is [0, 256].

**Note:**

Use the don't care data (256) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:CANFd:DATA?

**RESPONSE FORMAT**

<data>

<data>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data1 of the CAN FD bus trigger to 0x2E.

Command message:

*:TRIGger:CANFd:DATA 46*

*TRIG:CANF:DATA 46*

Query message:

*TRIG:CANF:DATA?*

Response message:

*46*

**RELATED COMMANDS**

:TRIGger:CANFd:CONDition

:TRIGger:CANFd:DAT2

**:TRIGger:CANFd:FTYPE****Command/Query****DESCRIPTION**

This command sets the frame type of the CAN FD bus trigger.

The query returns the current frame type of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:FTYPE <frame\_type>

<frame\_type>:= {BOTH|CAN|CANFd}

**QUERY SYNTAX**

:TRIGger:CANFd:FTYPE?

**RESPONSE FORMAT**

<frame\_type>

<frame\_type>:= {BOTH|CAN|CANFd}

**EXAMPLE**

The following command sets the frame type of the CAN FD bus trigger to CANFd.

Command message:

```
:TRIGger:CANFd:FTYPE CANFd  
TRIG:CANF:FTYP CANF
```

Query message:

```
TRIG:CANF:FTYP?
```

Response message:

```
CANFd
```

**:TRIGger:CANFd:ID****Command/Query****DESCRIPTION**

The command sets the ID of the CAN FD bus trigger when the trigger condition is Remote, ID or ID+Data.

The query returns the current ID of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:ID <id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

When the ID length is 29 bits, the range of the value is [0, 536870912] where 536870912 represents 0XXXXXXXXX. When the ID length is 11 bits, the range of the value is [0, 2048] where 2048 represents 0XXXXX.

**Note:**

Use the don't care data (536870912, ID length is 29) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:CANFd:ID?

**RESPONSE FORMAT**

<id>

<id>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the ID of the CAN FD trigger to 0x56A710C.

Command message:

```
:TRIGger:CANFd:ID 90861836  
TRIG:CANF:ID 90861836
```

Query message:

```
TRIG:CANF:ID?
```

Response message:

```
90861836
```

**RELATED COMMANDS**

```
:TRIGger:CANFd:CONDition  
:TRIGger:CANFd:IDLength
```

**:TRIGger:CANFd:IDLength****Command/Query****DESCRIPTION**

The command sets the ID length of the CAN FD bus trigger.

The query returns the current ID length of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:IDLENgth <length>

<length>:= {11BITS|29BITS}

**QUERY SYNTAX**

:TRIGger:CANFd:IDLENgth?

**RESPONSE FORMAT**

<length>

<length>:= {11BITS|29BITS}

**EXAMPLE**

The following command sets the ID length of the CAN FD bus trigger to 29BITS.

Command message:

*:TRIGger:CANFd:IDLength 29BITS*  
*TRIG:CANF:IDL 29BITS*

Query message:

*TRIG:CANF:IDL?*

Response message:

*29BITS*

**RELATED COMMANDS**

:TRIGger:CANFd:CONDition  
:TRIGger:CANFd:ID

**:TRIGger:CANFd:SOURce****Command/Query****DESCRIPTION**

The command selects the source of the CAN FD bus trigger.

The query returns the current source of the CAN FD bus trigger.

**COMMAND SYNTAX**

:TRIGger:CANFd:SOURce <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:CANFd:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the source of the CAN FD bus trigger as channel 2.

Command message:

*:TRIGger:CANFd:SOURce C2*

*TRIG:CANF:SOUR C2*

Query message:

*TRIG:CANF:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:CANFd:THReshold

**:TRIGger:CANFd:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on CAN FD bus triggering.

The query returns the current threshold of the source on CAN FD bus triggering.

**COMMAND SYNTAX**

:TRIGger:CANFd:THReshold <threshold>

<threshold>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:CANFd:THReshold?

**RESPONSE FORMAT**

<threshold>

<threshold>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the source on CAN FD bus trigger to 1.5 V.

Command message:

*:TRIGger:CANFd:THReshold 1.50E+00*  
*TRIG:CANF:THR 1.50E+00*

Query message:

*TRIG:CANF:THR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:CANFd:SOURce

**:TRIGger:IIS Commands [Option]**

The :TRIGGER:IIS subsystem commands control the IIS bus trigger parameters.

- ◆ **:TRIGger:IIS:AVARiant**
- ◆ **:TRIGger:IIS:BCLKSource**
- ◆ **:TRIGger:IIS:BCLKThreshold**
- ◆ **:TRIGger:IIS:BITOrder**
- ◆ **:TRIGger:IIS:CHANnel**
- ◆ **:TRIGger:IIS:COMPare**
- ◆ **:TRIGger:IIS:CONDition**
- ◆ **:TRIGger:IIS:DLENgth**
- ◆ **:TRIGger:IIS:DSource**
- ◆ **:TRIGger:IIS:DTHReshold**
- ◆ **:TRIGger:IIS:LATChedge**
- ◆ **:TRIGger:IIS:LCH**
- ◆ **:TRIGger:IIS:VALue**
- ◆ **:TRIGger:IIS:WSSource**
- ◆ **:TRIGger:IIS:WSTHreshold**



**:TRIGger:IS:AVARiant**

**Command/Query**

**DESCRIPTION**

The command sets the audio variant of the IIS bus trigger.

The query returns the current audio variant of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IS:AVARiant <type>

<type>:= {I2S|LJ|RJ}

**QUERY SYNTAX**

:TRIGger:IS:AVARiant?

**RESPONSE FORMAT**

<type>

<type>:= {I2S|LJ|RJ}

**EXAMPLE**

The following command sets the audio variant of the IIS bus trigger to IIS.

Command message:

*:TRIGger:IS:AVARiant I2S*  
*TRIG:IS:AVAR I2S*

Query message:

*TRIG:IS:AVAR?*

Response message:

*I2S*

**:TRIGger:IIS:BCLKSource****Command/Query****DESCRIPTION**

The command selects the BCLK source of the IIS bus trigger.

The query returns the current BCLK source of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:BCLKSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:IIS:BCLKSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the BCLK source of the IIS bus trigger as channel 2.

Command message:

*:TRIGger:IIS:BCLKSource C2*

*TRIG:IIS:BCLKS C2*

Query message:

*TRIG:IIS:BCLKS?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:IIS:BCLKThreshold

**:TRIGger:IS:BCLKThreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the BCLK on LIN bus trigger.

The query returns the current threshold of the BCLK on LIN bus trigger.

**COMMAND SYNTAX**

:TRIGger:IS:BCLKThreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:IS:BCLKThreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the BCLK on LIN bus trigger to 1.5 V.

Command message:

*:TRIGger:IS:BCLKThreshold 1.50E+00*  
*TRIG:IS:BCLKT 1.50+00*

Query message:

*TRIG:IS:BCLKT?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:IS:BCLKSource

**:TRIGger:IIS:BITorder****Command/Query****DESCRIPTION**

The command sets the bit order of the IIS bus trigger.

The query returns the current bit order of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:BITorder <order>

<order>:= {LSM|MSB}

**QUERY SYNTAX**

:TRIGger:IIS:BITorder?

**RESPONSE FORMAT**

<order>

<order>:= {LSM|MSB}

**EXAMPLE**

The following command sets the bit order of the IIS bus trigger to MSB.

Command message:

*:TRIGger:IIS:BITorder MSB*

*TRIG:IIS:BIT MSB*

Query message:

*TRIG:IIS:BIT?*

Response message:

*MSB*

**:TRIGger:IIS:CHANnel****Command/Query****DESCRIPTION**

The command sets the channel of the IIS bus trigger.

The query returns the current channel of the IIS bus trigger

**COMMAND SYNTAX**

:TRIGger:IIS:CHANnel <channel>

<channel>:= {LEFT|RIGHT}

**QUERY SYNTAX**

:TRIGger:IIS:CHANnel?

**RESPONSE FORMAT**

<channel>

<channel>:= {LEFT|RIGHT}

**EXAMPLE**

The following command sets to trigger on right channel of the IIS bus.

Command message:

*:TRIGger:IIS:CHANnel RIGHT*

*TRIG:IIS:CHAN RIGHT*

Query message:

*TRIG:IIS:CHAN?*

Response message:

*RIGHT*

**:TRIGger:IIS:COMPare****Command/Query****DESCRIPTION**

The command sets the data compare type of the IIS bus trigger.

The query returns the current data compare type of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:COMPare <type>

<type>:= {EQUal|GREaterthan|LESSthan}

**QUERY SYNTAX**

:TRIGger:IIS:COMPare?

**RESPONSE FORMAT**

<type>

<type>:= {EQUal|GREaterthan|LESSthan}

**EXAMPLE**

The following command sets the data compare type of the IIS bus trigger to LESSthan.

Command message:

*:TRIGger:IIS:COMPare LESSthan*

*TRIG:IIS:COMP LESS*

Query message:

*TRIG:IIS:COMP?*

Response message:

*LESSthan*

**RELATED COMMANDS**

:TRIGger:IIS:CONDition

**:TRIGger:IIS:CONDition****Command/Query****DESCRIPTION**

The command sets the trigger condition of the IIS bus.

The query returns the current trigger condition of the IIS bus.

**COMMAND SYNTAX**

:TRIGger:IIS:CONDition <condition>

<condition>:= {DATA|MUTE|CLIP|GLITch|RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:IIS:CONDition?

**RESPONSE FORMAT**

<condition>

<condition>:= {DATA|MUTE|CLIP|GLITch|RISing|FALLing}

**EXAMPLE**

The following command sets the trigger condition of the IIS bus to DATA.

Command message:

*:TRIGger:IIS:CONDition DATA*

*TRIG:IIS:COND DATA*

Query message:

*TRIG:IIS:COND?*

Response message:

*DATA*

**:TRIGger:IIS:DLENgth****Command/Query****DESCRIPTION**

The command sets the data bits of the IIS bus trigger.

The query returns the current data bits of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:DLENgth <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

The range of the value is related to the channel bits and the start bits. If the channel bits are 32 and the start bit is 2, the range is [1,30]

**QUERY SYNTAX**

:TRIGger:IIS:DLENgth?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the data bits of the IIS bus trigger to 10.

Command message:

*:TRIGger:IIS:DLENgth 10*

*TRIG:IIS:DLEN 10*

Query message:

*TRIG:IIS:DLEN?*

Response message:

*10*



**:TRIGger:IIS:DSource****Command/Query****DESCRIPTION**

The command selects the data source of the IIS bus trigger.

The query returns the current data source of the IIS bus trigger

**COMMAND SYNTAX**

**:TRIGger:IIS:DSource <source>**

**<source>:= {C<n>|D<d>}**

**<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.**

**<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.**

**QUERY SYNTAX**

**:TRIGger:IIS:DSource?**

**RESPONSE FORMAT**

**<source>**

**<source>:= {C<n>|D<d>}**

**EXAMPLE**

The following command sets the data source of the IIS bus trigger as C2.

Command message:

*:TRIGger:IIS:DSource C2*

*TRIG:IIS:DS C2*

Query message:

*TRIG:IIS:DS?*

Response message:

*C2*

**RELATED COMMANDS**

**:TRIGger:IIS:DTHReshold**

**:TRIGger:IIS:DTHReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the data source on IIS bus trigger.

The query returns the current threshold of the data source on IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:DTHReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:IIS:DTHReshold?

**RESPONSE FORMAT**

<threshold>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the data source on IIS bus trigger to 1.5 V.

Command message:

*TRIGger:IIS:DTHReshold 1.50E+00*

*TRIG:IIS:DTHR 1.50E+00*

Query message:

*TRIG:IIS:DTHR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:IIS:DSource

**:TRIGger:IIS:LATChedge****Command/Query****DESCRIPTION**

The command selects the sampling edge of BCLK on IIS bus trigger.

The query returns the sampling edge of BCLK on IIS bus trigger

**COMMAND SYNTAX**

:TRIGger:IIS:LATChedge <slope>

<slope>:= {RISing|FALLing}

**QUERY SYNTAX**

:TRIGger:IIS:LATChedge?

**RESPONSE FORMAT**

<slope>

<slope>:= {RISing|FALLing}

**EXAMPLE**

The following command sets the sampling edge of BCLK on IIS bus trigger to RISing.

Command message:

*:TRIGger:IIS:LATChedge RISing*  
*TRIG:IIS:LATC RIS*

Query message:

*TRIG:IIS:LATC?*

Response message:

*RISing*

**:TRIGger:IIS:LCH****Command/Query****DESCRIPTION**

The command selects the level of the left channel on IIS bus trigger.

The query returns the current level of the left channel on IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:LCH <level>

<level>:= {LOW|HIGH}

**QUERY SYNTAX**

:TRIGger:IIS:LCH?

**RESPONSE FORMAT**

<level>

<level>:= {LOW|HIGH}

**EXAMPLE**

The following command sets the level of the left channel on IIS bus trigger to HIGH.

Command message:

*:TRIGger:IIS:LCH HIGH*

*TRIG:IIS:LCH HIGH*

Query message:

*TRIG:IIS:LCH?*

Response message:

*HIGH*

**:TRIGger:IIS:VALue****Command/Query****DESCRIPTION**

The command sets the value of the IIS bus trigger.

The query returns the current value of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:VALue <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

- The range of the value is related to data length by using the command :TRIGger:IIS:DLENgth.
- Use the don't care data (256, data length is 8) to ignore the data value.

**QUERY SYNTAX**

:TRIGger:IIS:VALue?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the value of the IIS bus trigger to 0x56 when the data length is 8.

Command message:

*:TRIGger:IIS:VALue 86*

*TRIG:IIS:VAL 86*

Query message:

*TRIG:IIS:VAL?*

Response message:

*86*

**RELATED COMMANDS**

:TRIGger:IIS:CONDition

:TRIGger:IIS:DLENgth

**:TRIGger:IIS:WSSource****Command/Query****DESCRIPTION**

The command selects the WS source of the IIS bus trigger.

The query returns the current WS source of the IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:WSSource <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:IIS:WSSource?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the WS source of the IIS bus trigger as channel 2.

Command message:

```
:TRIGger:IIS:WSSource C2
TRIG:IIS:WSS C2
```

Query message:

```
TRIG:IIS:WSS?
```

Response message:

```
C2
```

**RELATED COMMANDS**

:TRIGger:IIS:WSTHreshold

**:TRIGger:IIS:WSTHreshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the WS on IIS bus trigger.

The query returns the current threshold of the WS on IIS bus trigger.

**COMMAND SYNTAX**

:TRIGger:IIS:WSTHreshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | $[-4.26 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.26 \times \text{vertical\_scale} - \text{vertical\_offset}]$ |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | $[-4.5 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.5 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | $[-4.1 \times \text{vertical\_scale} - \text{vertical\_offset}, 4.1 \times \text{vertical\_scale} - \text{vertical\_offset}]$   |

**QUERY SYNTAX**

:TRIGger:IIS:WSTHreshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the WS on IIS bus trigger to 1.5 V.

Command message:

*:TRIGger:IIS:WSTHreshold 1.50E+00  
TRIG:IIS:WSTH 1.50E+00*

Query message:

*TRIG:IIS:WSTH?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:IIS:WSSource

**:TRIGger:SENT Commands [Option]**

The :TRIGGER:SENT subsystem commands control the SENT bus trigger parameters.

- ◆ **:TRIGger:SENT:SOURce**
- ◆ **:TRIGger:SENT:THReshold**



**:TRIGger:SENT:SOURce****Command/Query****DESCRIPTION**

The command selects the source of the SENT bus trigger.

The query returns the current source of the SENT bus trigger.

**COMMAND SYNTAX**

:TRIGger:SENT:Source <source>

<source>:= {C<n>|D<d>}

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:TRIGger:SENT:Source?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|D<d>}

**EXAMPLE**

The following command sets the source of the SENT bus trigger as channel 2.

Command message:

*:TRIGger:SENT:SOURce C2*  
*TRIG:SENT:SOUR C2*

Query message:

*TRIG:SENT:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:TRIGger:SENT:THReshold:ACQuire:SEQuence

**:TRIGger:SENT:THReshold**

**Command/Query**

**DESCRIPTION**

The command sets the threshold of the source on SENT bus trigger.

The query returns the current threshold of the source on SENT bus trigger.

**COMMAND SYNTAX**

:TRIGger:SENT:THReshold <value>

<value>:= Value in NR3 format, including a decimal point and exponent, like 1.23E+2.

The range of the value varies by model, see the table below for details.

| Model   | Value Range   |
|---|---|
| SDS7000A  | [-4.26*vertical_scale-vertical_offset, 4.26*vertical_scale-vertical_offset] |
| SDS6000 Pro<br>SDS6000A<br>SDS6000L                     | [-4.5*vertical_scale-vertical_offset, 4.5*vertical_scale-vertical_offset]   |
| SDS5000X<br>SDS3000X HD<br>SDS2000X Plus<br>SDS2000X HD | [-4.1*vertical_scale-vertical_offset, 4.1*vertical_scale-vertical_offset]   |

**QUERY SYNTAX**

:TRIGger:SENT:THReshold?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR3 format.

**EXAMPLE**

The following command sets the threshold of the source on FLEXray bus trigger to 1.5 V.

Command message:

*:TRIGger:SENT:THReshold 1.50E+00*  
*TRIG:SENT:THR 1.50E+00*

Query message:

*TRIG:SENT:THR?*

Response message:

*1.50E+00*

**RELATED COMMANDS**

:TRIGger:SENT:Source

## WAVEform Commands

The WAVEFORM subsystem is used to transfer data to a controller from the oscilloscope waveform memory.

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. 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.

- ◆ **:WAVEform:BYTeorder**
- ◆ **:WAVEform:STARt**
- ◆ **:WAVEform:INTerval**
- ◆ **:WAVEform:MAXPoint**
- ◆ **:WAVEform:POINt**
- ◆ **:WAVEform:PREamble**
- ◆ **:WAVEform:SEQuence**
- ◆ **:WAVEform:SOURce**
- ◆ **:WAVEform:STARt**
- ◆ **:WAVEform:WIDTh**

**:WAVeform:BYTeorder****Command/Query****DESCRIPTION**

The command sets the byte order of transmitted waveform data words. When waveform data is transmitted in 16 bits, byte order needs to be set, defaulting to LSB.

The query returns the transmission order of waveform data.

**COMMAND SYNTAX**

:WAVeform:BYTeorder <order>

<order>:= {MSB|LSB}

**QUERY SYNTAX**

:WAVeform:BYTeorder?

**RESPONSE FORMAT**

<order>

<order>:= {MSB|LSB}

**EXAMPLE**

The following command sets the byte order of data words transmitted by the oscilloscope to MSB

Command message:

*:WAVeform:BYTeorder MSB*

*WAV:BYT MSB*

Query message:

*WAV:BYT?*

Response message:

*MSB*

**RELATED COMMANDS**

:WAVeform:WIDTh

**:WAVeform:SOURce****Command/Query****DESCRIPTION**

The command specifies the source waveform to be transferred from the oscilloscope using the query :WAVeform:DATA?

The query returns the source waveform to be transferred from the oscilloscope.

**COMMAND SYNTAX**

:WAVeform:SOURce <source>

<source>:= {C<n>|F<x>|D<d>}

- ◆ C denotes an analog channel.
- ◆ F denotes a math function.
- ◆ D denotes a digital channel.

<n>:= 1 to (# analog channels) in NR1 format, including an integer and no decimal point, like 1.

<x>:= 1 to (# math functions) in NR1 format, including an integer and no decimal point, like 1.

<d>:= 0 to (# digital channels - 1) in NR1 format, including an integer and no decimal point, like 1.

**QUERY SYNTAX**

:WAVeform:SOURce?

**RESPONSE FORMAT**

<source>

<source>:= {C<n>|F<x>|D<d>}

**EXAMPLE**

The following command specifies that the Channel 2 waveform will be transferred in the next :WAVeform:DATA? query or :WAVeform:PREamble? query.

Command message:

*:WAVeform:SOURce C2*  
*WAV:SOUR C2*

Query message:

*WAV:SOUR?*

Response message:

*C2*

**RELATED COMMANDS**

:WAVeform:DATA  
:WAVeform:PREamble

**:WAVeform:STARt****Command/Query****DESCRIPTION**

The command specifies the starting data point for waveform transfer using the query :WAVeform:DATA?.

The query returns the starting data point for waveform transfer.

**COMMAND SYNTAX**

:WAVeform:STARt <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

The value range is related to the current waveform point and the value set by the command :WAVeform:POINT.

**QUERY SYNTAX**

:WAVeform:STARt?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the start point to 1000 when the current waveform point is 400 kpts.

Command message:

```
:WAVeform:STARt 1000  
WAV:STAR 1000
```

Query message:

```
WAV:STAR?
```

Response message:

```
1000
```

**RELATED COMMANDS**

:WAVeform:POINT

**:WAVeform:INTerval****Command/Query****DESCRIPTION**

The command sets the interval between data points for waveform transfer using the query :WAVeform:DATA?

The query returns the interval between data points for waveform transfer.

**COMMAND SYNTAX**

:WAVeform:INTerval <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

The value range is related to the values set by the command :WAVeform:POINT and :WAVeform:START.

**QUERY SYNTAX**

:WAVeform:INTerval?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command sets the interval between data points for waveform transfer to 200.

Command message:

*:WAVeform:INTerval 200*  
*WAV:INT 200*

Query message:

*WAV:INT?*

Response message:

*200*

**RELATED COMMANDS**

:WAVeform:START  
:WAVeform:POINT

**:WAVeform:POINt****Command/Query****DESCRIPTION**

The command sets the number of waveform points to be transferred with the query :WAVeform:DATA?

The query returns the number of waveform points to be transferred.

**COMMAND SYNTAX**

:WAVeform:POINt <value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**Note:**

The value range is related to the current waveform point.

**QUERY SYNTAX**

:WAVeform:POINt?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following command the number of waveform points to be transferred to 2000 pts.

Command message:

*:WAVeform:POINt 20000*  
*WAV:POIN 20000*

Query message:

*WAV:POIN?*

Response message:

*20000*

**RELATED COMMANDS**

:ACQuire:POINts



**:WAVeform:MAXPoint****Query****DESCRIPTION**

The query returns the maximum points of one piece, when it needs to read the waveform data in pieces.

**QUERY SYNTAX**

:WAVeform:MAXPoint?

**RESPONSE FORMAT**

<value>

<value>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

The following return the maximum points of one piece in SDS2000X Plus series.

Query message:

*:WAV:MAXP?*

Response message:

*1000000*

**:WAVeform:WIDTh****Command/Query****DESCRIPTION**

The command sets the current output format for the transfer of waveform data.

The query returns the current output format for the transfer of waveform data.

**COMMAND SYNTAX**

:WAVeform:WIDTh <type>

<type>:= {BYTE|WORD}

WORD formatted data transfers 16-bit data as two bytes, and transmission in LSB format

BYTE formatted data is transferred as 8-bit bytes.

**Note:**

When the vertical resolution is set to 10 bit or the ADC bit is more than 8bit, it must to use the command to set to WORD before transferring waveform data.

**QUERY SYNTAX**

:WAVeform:WIDTh?

**RESPONSE FORMAT**

<type>

<type>:= {BYTE|WORD}

**EXAMPLE**

The following command sets the current output format for the transfer of waveform data to BYTE.

Command message:

*:WAVeform:WIDTh BYTE*  
*WAV:WIDTh BYTE*

Query message:

*WAV:WIDTh?*

Response message:

*BYTE*

**:WAVeform:PREamble****Query****DESCRIPTION**

The query returns the parameters of the source using by the command :WAVeform:SOURce.

**QUERY SYNTAX**

:WAVeform:PREamble?

**RESPONSE FORMAT**

<bin>

<bin>:= binary data block headed " #9<9-Digits>". See the table below for details.

**RELATED COMMANDS**

:WAVeform:SOURce

**Table 1 Explanation of the descriptor block**

(The first byte after "#9<9-digits>" is the starting position)

| Addresses | Type  | Length | Description   |
|-----------|-------|--------|---|
| 0~15      | char  | 16     | Descriptor name. It is string, the first 8 chars are always "WAVEDESC"  |
| 16~31     | char  | 16     | Template name. It is string, the first 7 chars are always "WAVEACE".  |
| 32~33     | short | 2      | COMM_TYPE. It is chosen by remote command comm_format. 0 - byte, 1- word. Default value is 0.   |
| 34~35     | short | 2      | COMM_ORDER. It is chosen by remote command comm_format. 0 - LSB, 1- MSB. Default value is 0.  |
| 36~39     | long  | 4      | wave_desc_length. Length in bytes of block WAVEDESC. (346)  |
| 40~59     | long  | 4*5    | Reserved  |
| 60~63     | long  | 4      | WAVE_ARRAY_1. Length in bytes of 1st simple data array. In transmitted waveform, represent the number of transmitted bytes in accordance with the parameter of the :WAVeform:WIDTh remote command and the used format (see COMM_TYPE). Only for analog channel. |
| 64~75     | long  | 4*3    | Reserved  |
| 76~91     | char  | 16     | Instrument name. It is string, always "Siglent SDS".  |
| 92~95     | long  | 4      | Reserved  |
| 96~111    | char  | 16     | Reserved  |
| 112~115   | long  | 4      | Reserved  |
| 116~119   | long  | 4      | Wave array count. Number of data points in the data array. Only for analog channel. When sequence is on, this value means the points number of single sequence frame.   |
| 120~131   | long  | 4*3    | Reserved  |
| 132~135   | long  | 4      | First point. Indicates the offset relative to the beginning of the trace buffer. Value is the same as the parameter of the :WAVeform:STARt remote command.  |
| 136~139   | long  | 4      | Data interval. Indicates the interval between data points for waveform transfer. Value is the same as the parameter of the :WAVeform:INTerval remote command.   |
| 140~143   | long  | 4      | Reserved  |

|         |             |     |  |
|---------|-------------|-----|--|
| 144~147 | long        | 4   | read_frames, number of sequence frames transferred this time. Used to calculate the reading times of sequence waveform   |
| 148~151 | long        | 4   | sum_frames, number of sequence frames acquired. Used to calculate the reading times of sequence waveform   |
| 152~155 | short       | 2*2 | Reserved   |
| 156~159 | float       | 4   | Vertical gain. The value of vertical scale without probe attenuation.  |
| 160~163 | float       | 4   | Vertical offset. The value of vertical offset without probe attenuation.   |
| 164~167 | float       | 4   | code_per_div. The value is different for different vertical gain of different models   |
| 168~171 | float       | 4   | Reserved   |
| 172~173 | short       | 2   | Adc_bit  |
| 174~175 | short       | 2   | The specified frame index of sequence set by the parameter <value1> of the command :WAVEform:SEquence. Default Value is 1  |
| 176~179 | float       | 4   | Horizontal interval. Sampling interval for time domain waveforms. Horizontal interval = 1/sampling rate.   |
| 180~187 | long double | 8   | Horizontal offset. Trigger offset for the first sweep of the trigger, seconds between the trigger and the first data point. Unit is s.                                 |
| 188~195 | long double | 8   | Reserved   |
| 196~243 | char        | 48  | Reserved   |
| 244~291 | char        | 48  | Reserved   |
| 292~295 | float       | 4   | Reserved   |
| 296~311 | struct      | 16  | Reserved   |
| 312~315 | float       | 4   | Reserved   |
| 316~323 | short       | 2*4 | Reserved   |
| 324~325 | short       | 2   | Time_base. This is the enumerated time/div. see the Table 2 for details.   |
| 326~327 | short       | 2   | Vertical coupling. 0-DC,1-AC,2-GND   |
| 328~331 | float       | 4   | Probe attenuation.   |
| 332~333 | short       | 2   | Fixed vertical gain. This is the enumerated vertical scale. This value is not intuitive, and the vertical scale is usually represented by the value of address 156~159 |
| 334~335 | short       | 2   | Bandwidth limit. 0-OFF,1-20M,2-200M  |
| 336~343 | float       | 4*2 | Reserved   |
| 344~345 | short       | 2   | Wave source. 0-C1,1-C2,2-C3,3-C4,4-C5,5-C6,6-C7,7-C8   |

**Table 2 Enum of Timebase**

| Index | Timebase (s) | Index | Timebase(s) |
|-------|--------------|-------|-------------|
| 0     | 200E-12      | 20    | 1E-3        |
| 1     | 500E-12      | 21    | 2E-3        |
| 2     | 1E-9         | 22    | 5E-3        |
| 3     | 2E-9         | 23    | 10E-3       |
| 4     | 5E-9         | 24    | 20E-3       |
| 5     | 10E-9        | 25    | 50E-3       |
| 6     | 20E-9        | 26    | 100E-3      |
| 7     | 50E-9        | 27    | 200E-3      |
| 8     | 100E-9       | 28    | 500E-3      |
| 9     | 200E-0       | 29    | 1           |
| 10    | 500E-9       | 30    | 2           |
| 11    | 1E-6         | 31    | 5           |
| 12    | 2E-6         | 32    | 10          |
| 13    | 5E-6         | 33    | 20          |
| 14    | 10E-6        | 34    | 50          |
| 15    | 20E-6        | 35    | 100         |
| 16    | 50E-6        | 36    | 200         |
| 17    | 100E-6       | 37    | 500         |
| 18    | 200E-6       | 38    | 1000        |
| 19    | 500E-6       |       |             |

**Note: Different models have different time base enumeration**

**:WAVeform:DATA****Query****DESCRIPTION**

The query returns the waveform data of the source using by the command :WAVeform:SOURce to be transferred from the oscilloscope.

**QUERY SYNTAX**

:WAVeform:DATA?

**RESPONSE FORMAT**

<wave\_data>

<wave\_data>:=binary data block headed " #N<N-Digits>"

**RELATED COMMANDS**

:WAVeform:STARt  
:WAVeform:INTerval  
:WAVeform:POINT  
:WAVeform:MAXPoint  
:WAVeform:WIDTh

**EXAMPLE**

For SDS5000X series, the following steps show how to use the command to reconstitute the display of waveform.

For analog channel waveform and math waveform (except for FFT):

**Step 1: Send the commands to get the data of waveform.**

Command message:

:WAVeform:SOURce C2

:WAVeform:DATA?

Response message:

The header is “#9000001000” which nine ASCII integers are used to give the number of the waveform data points (1000 pts). After the header of block, is beginning of the wave data, and the last two bytes “0A 0A” means the end of data.

| Data   | Description      |
|--|------------------|
| -----  | -----            |
| 23 39 30 30 30 30 31 30 30 30 F5 F6 F6 F6 F6 F7 F7 F8    | #9000001000..... |
| F8 F9 F9 F9 FA FA FB FB FB FC FC FC FD FE FF FF 00 01 01 | .....            |
| 02 02 03 03 04 05 06 06 06 07 08 09 09 0A 0A 0B 0B 0C 0D | .....            |
| 0D 0E 0F 0F 10 11 12 13 14 14 15 15 16 17 18 19 1A 1B 1B | .....            |
| 1B 1C 1C 1E 1F 1F 20 21 22 22 23 24 25 ..... ED ED ED ED | .....            |
| ED ED ED ED ED ED ED ED ED EE EE EE ED ED ED EE EE EF    | .....            |
| EF EF EE EE EF EF F0 F0 F0 F1 F1 F1 F1 F1 F2 F2 F3 F3 F3 | ....\n\n         |
| F3 F3 F4 F4 0A 0A  |                  |

**Step 2: Send the query to get the parameters of waveform.**

Query message:

```
:WAVeform:PRE?
```

For parameter parsing, see the section of the query. Through the query, we can get the vertical scale is 10 V/div, the vertical offset is 14.5 V, the timebase is 20E-9 s, the trigger delay is 1.72E-8 s, and the sampling interval is 2E-10 s.

**Step 3: Calculate the voltage value corresponding to the data point.**

Using the formula: voltage value (V) = code value \*(vdiv /code\_per\_div) - voffset.

| Parameter    | Description   | Example above  |
|--------------|---|--|
| code value   | Signed number of wave data. If the vertical resolution of the model is greater than 8bit, the code value is a word in LSB byte order by the command :WAVeform:WIDTh. The data is left aligned, and the lower bit is zero filled.<br><br>For SDS6000 Pro, 12bit data is used for 12bit and 10bit models, but the ADC range is different. | The first point is the 17th data “F5”, convert to decimal is “-11” |
| vdiv         | The vertical scale.<br>It is the value with address 156~159 in the data block returned by the :WAVeform:PREamble?   | 10   |
| voffset      | The vertical position value.<br>It is the value with address 160~163 in the data block returned by the :WAVeform:PREamble?  | 14.5   |
| code_per_div | Code value per division in vertical direction.<br>It is the value with address 164~167 in the data block returned by the :WAVeform:PREamble?  | 30   |

The picture above as an example:

The first point: voltage value =  $-11*(10/30)-(14.5) = -18.167$  V.

**Step 4: Calculate the time value of the data point.**

Using the formula:  $\text{time value(S)} = \text{delay} - (\text{timebase} * \text{grid} / 2) + \text{index} * \text{interval}$

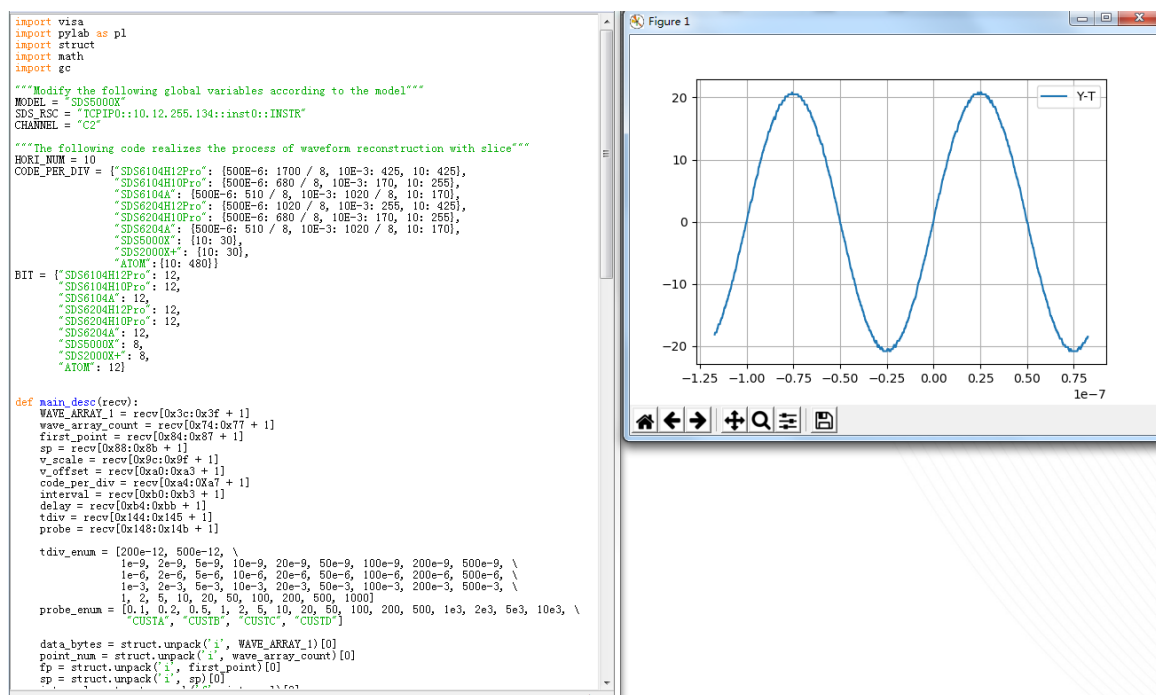
| Parameter | Description   | Example above |
|-----------|---|---------------|
| timebase  | The horizontal scale.<br>It is the value with address 324~325 in the data block returned by the :WAVEform:PREamble?   | 2E-8          |
| delay     | The horizontal position value.<br>It is the value with address 180~187 in the data block returned by the :WAVEform:PREamble?  | 1.72E-8       |
| grid      | The grid numbers in horizontal direction.<br>SDS7000A/SDS6000 Pro/SDS6000A/SDS5000X/<br>SDS3000X HD/SDS2000X Plus/SDS2000X<br>HD/SDS1000X HD/SDS800X HD:10<br>SHS800X/SHS1000X:12 | 10            |
| index     | The index of the data. The first point is 0.  | --            |
| interval  | Sampling interval.<br>It is the value with address 176~179 in the data block returned by the :WAVEform:PREamble?  | 2E-10         |

The picture above as an example:

The first data point:  $\text{time value} = (1.72E-8) - (2E-08 * 10 / 2) = -8.28E-08 \text{ s} = -82.8 \text{ ns}$ .

The second data point:  $\text{time value} = -82.8 \text{ ns} + 0.2\text{ns} = -82.6 \text{ ns}$ .

Use python to reconstruct the waveform: (See the code in Read Waveform Data Example)



For digital channel waveform:





**Step 1: Send the commands to get the data of waveform.**

Command message:

```
:WAVeform:SOURce D0
:WAVeform:DATA?
```

Response message:

The header is “#9000002500” which nine ASCII integers are used to give the number of the waveform data points (2500 pts). After the header of block, is beginning of the wave data. For digital, one bit represents a data point, if the number of points is not an integer multiple of 8, the byte less than 8 bits will be filled with 0. So there are 313 bytes. The last two bytes “0A 0A” means the end of data.

| Data   | Description      |
|--|------------------|
| 23 39 30 30 30 30 30 32 35 30 30 FF FF FF FF FF FF FF FF | #9000002500..... |
| FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    | .....            |
| FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00    | .....            |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .....            |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .....            |
| 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .....            |
| FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    | .....            |
| FF 0A 0A   | .\n\n            |

**Step 2: Send the query to get the parameters of waveform.**

Query message:

```
:WAVeform:PRE?
```

For parameter parsing, see the section of the query. Through the query, we can get the timebase is  $2E-7$  s, the trigger delay is  $2E-7$  s, and the sampling interval is  $2E-10$  s.

**Step 3: Covert to the high (1) and low (0) corresponding to the data point.**

According to the wave data, we can know the first eight points of waveform is the 17th byte “FF”, convert to binary is “11111111” (Hexadecimal converted to binary (LSB)).

**Step 4: Calculate the time value of the data point.**

Using the formula:  $\text{time value(S)} = \text{delay} - (\text{timebase} * \text{grid} / 2) + \text{index} * \text{interval}$

| Parameter | Description   | Example above |
|-----------|---|---------------|
| timebase  | The horizontal scale.   | $2E-7$        |
| delay     | The voltage position value.   | $2E-7$        |
| grid      | The grid numbers in horizontal direction.<br>SDS7000A/SDS6000Pro/SDS6000A/SDS5000X<br>/SDS300X HD/SDS2000X Plus/SDS2000X<br>HD/SDS1000X HD/SDS800X HD:10<br>SHS800X/SHS1000X:12 | 10            |
| index     | The index of the data. The first point is 0.  | --            |
| interval  | Sampling interval.  | $8E-10$       |

The picture above as an example:

The first data point:  $\text{time value} = 2E-7 - (2E-7 * 10 / 2) = -8E-07 \text{ s} = -800 \text{ ns}$ .

The second data point:  $\text{time value} = -800 \text{ ns} + 0.8 \text{ ns} = -799.2 \text{ ns}$ .

Use python to reconstruct the waveform: (See the code in Read Waveform Data of Digital Example)

```

import visa
import pylab as pl
import struct

def get_char_bit(char,n):
    return (char >> n) & 1

def main_desc(recv):
    first_point = recv[0x84:0x87+1]
    sp = recv[0x88:0x8b+1]
    interval = recv[0xb0:0xb3+1]
    delay = recv[0xb4:0xbb+1]
    tdiv = recv[0x144:0x145+1]
    tdiv_enum=[200e-12,500e-12,\
              1e-9,2e-9,5e-9,10e-9,20e-9,50e-9,100e-9,200e-9,500e-9,\
              1e-6,2e-6,5e-6,10e-6,20e-6,50e-6,100e-6,200e-6,500e-6,\
              1e-3,2e-3,5e-3,10e-3,20e-3,50e-3,100e-3,200e-3,500e-3,\
              1,2,5,10,20,50,100,200,500,1000]

    fp = struct.unpack('i',first_point)[0]
    sp = struct.unpack('i',sp)[0]
    interval = 0.2e-9*struct.unpack('f',interval)[0]
    delay = struct.unpack('d',delay)[0]
    tdiv_index = struct.unpack('h',tdiv)[0]
    tdiv = tdiv_enum[tdiv_index]
    return interval,delay,tdiv

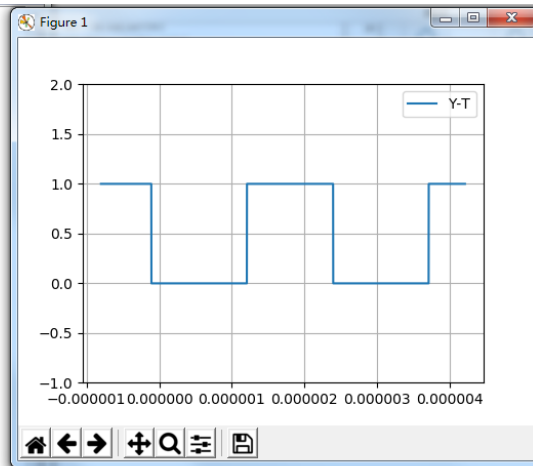
def main_new_scpi():
    _rm = visa.ResourceManager()
    sds = _rm.open_resource("TCPIP0::10.12.255.134::inst0::INSTR")
    sds.write("WAV:SOUR D8")
    sds.write("WAV:PREAmble?")
    recv_all = sds.read_raw()
    recv = recv_all[recv_all.find(b'#')+11:]
    interval,trdl,tdiv = main_desc(recv)
    print("interval=[0],trdl=[1],tdiv=[2]".format(interval,trdl,tdiv))
    sds.write("WAV:DATA?")
    recv_rtn = sds.read_raw()
    recv = list(recv_rtn[recv_rtn.find(b'#')+11:-2])
    volt_value = []
    data = bytearray(recv)

    for char in data:
        for i in range(0,8):
            volt_value.append(get_char_bit(char,i))
    print(len(volt_value))
    time_value = []
    for idx in range(0,len(volt_value)):
        time_data = -float(trdl)-(float(tdiv)*10/2)+idx*interval
        time_value.append(time_data)

    pl.figure(figsize=(7,5))
    pl.ylim(-1,2)
    pl.plot(time_value,volt_value,markersize=2,label="Y-T")
    pl.legend()
    pl.grid()
    pl.show()

if __name__=="__main__":
    main_new_scpi()

```



For an example of FFT waveform, please refer to Read Waveform Data of FFT Example.

**:WAVeform:SEQuence****Command/Query****DESCRIPTION**

This command is used to set the sequence waveform frame to be read. Valid only when sequence is on.

The query returns the index of sequence frame to be transferred.

**COMMAND SYNTAX**

:WAVeform:SEQuence <value1>,<value2>

<value1>:= Value in NR1 format, including an integer and no decimal point, like 1.

It sets the index of sequence frame to be transferred with the query :WAVeform:DATA?. When set to 0, all sequence frames are returned and the query :WAVeform:DATA? will transfer as much sequence frames as it can transfer at once.

<value2>:= Value in NR1 format, including an integer and no decimal point, like 1.

It sets the start index of sequence frame to be transferred with the query :WAVeform:DATA? This value is valid when <value1> is set to 0.

Due to the memory limitation, when the number of all frames exceeds the limit, it is necessary to read by slice through <value2>. The number of slices can be calculated by read\_frames (0x90-0x93) and sum\_frames(0x94-0x97) in the query :WAVeform:PREamble?

**Note:**

- When sequence is enabled, <value1> will be set to 1 by default;when sequence and history are enabled, <value1> will be set to the current frame by default. In other cases, <value1> is set to 4294967295 by default.
- The value range is related to the current sequence number.

**QUERY SYNTAX**

:WAVeform:SEQuence?

**RESPONSE FORMAT**

<value1>,<value2>

<value1>:= Value in NR1 format, including an integer and no decimal point, like 1.

<value2>:= Value in NR1 format, including an integer and no decimal point, like 1.

**EXAMPLE**

After 5 frames are acquired in sequence mode, and history is enabled, there are 10 kpts per frame, totaling 50 kpts. After sending the following command, all segments can be read at one time through the query :WAVeform:DATA?.

Command message:

```
:WAVeform:SEQuence 0,1
WAV:SEQ 0,1
```

Query message:

```
WAV:SEQ?
```

Response message:

```
0,1
```

See the python code in Read Sequence Waveform Data Example for reference.

## WGEN Commands

When the oscilloscope supports the function generator module (built-in waveform generator or SAG1021I) and is licensed (Option FG), you can output sine, square, ramp, pulse, DC, noise, exponential rise, exponential fall, cardiac, gaussian pulse and arbitrary waveforms. The WGEN commands are used to select the waveform function and parameters.

The WGEN commands are the same as that of Siglent SDG series, so the format is not consistent with other groups. Refer to SDG programming guide for details.

- ◆ **ARbWaVe**
- ◆ **BaSic\_WaVe**
- ◆ **OUTPut**
- ◆ **SToreList**
- ◆ **SYNC**
- ◆ **VOLTPRT**

## ARbWaVe

### Command/Query

#### DESCRIPTION

This command sets or gets the basic wave parameters.

#### COMMAND SYNTAX

<channel>:ARbWaVe INDEX,<index>

<channel>:ARbWaVe NAME,<name>

<channel>:={C1}, SAG and the built-in waveform generator only support one output channel.

<index>:= the index of the arbitrary waveform from the table below.

<name>:= the name of the arbitrary waveform from the table below.

#### Note:

This table is just an example, the index depends on the specific model. The “STL?” query can be used to get the accurate mapping relationship between the index and name.

| Index | Name     | Index | Name      | Index | Name      | Index | Name     |
|-------|----------|-------|-----------|-------|-----------|-------|----------|
| 0     | Sine     | 12    | Logfall   | 24    | Gmonopuls | 36    | Triang   |
| 1     | Noise    | 13    | Logrise   | 25    | Tripuls   | 37    | Harris   |
| 2     | StairUp  | 14    | Sqrt      | 26    | Cardiac   | 38    | Bartlett |
| 3     | StairDn  | 15    | Root3     | 27    | Quake     | 39    | Tan      |
| 4     | Stairud  | 16    | X^2       | 28    | Chirp     | 40    | Cot      |
| 5     | Ppulse   | 17    | X^3       | 29    | Twotone   | 41    | Sec      |
| 6     | Npulse   | 18    | Sinc      | 30    | Snr       | 42    | Csc      |
| 7     | Trapezia | 19    | Gaussian  | 31    | Hamming   | 43    | Asin     |
| 8     | Up ramp  | 20    | Dlorentz  | 32    | Hanning   | 44    | Acos     |
| 9     | Dn ramp  | 21    | Haversine | 33    | Kaiser    | 45    | Atan     |
| 10    | Exp_fall | 22    | Lorentz   | 34    | Blackman  | 46    | Acot     |
| 11    | Exp_rise | 23    | Gauspuls  | 35    | Gausswin  | 47    | Square   |

#### QUERY SYNTAX

<channel>:ARbWaVe?

<channel>:={C1}

#### RESPONSE FORMAT

<channel>:ARWV

INDEX,<index>,NAME,<name>

#### RELATED COMMANDS

SToreList

**EXAMPLE**

Set CH1 current waveform by index 2:

*C1:ARWV INDEX,2*

Read CH1 current waveform:

*C1:ARWV?*

Return:

*C1:ARWV INDEX,2,NAME,StairUp*

Set CH1 current waveform to wave\_1 by name.

*C1:ARWV NAME,wave\_1*

## BaSic\_WaVe

### Command/Query

#### DESCRIPTION

This command sets or gets the basic wave parameters.

#### COMMAND SYNTAX

<channel>:BaSic\_WaVe <parameter>,<value>

<channel>:={C1}, SAG and the built-in waveform generator only support one output channel.

<parameter>:= a parameter from the table below.

<value>:= value of the corresponding parameter.

| Parameters | Value       | Description   |
|------------|-------------|---|
| WVTP       | <type>      | := {SINE, SQUARE, RAMP, PULSE, NOISE, ARB, DC, PRBS, IQ}. If the command doesn't set basic waveform type, WVTP will be set to the current waveform. |
| FRQ        | <frequency> | := frequency. The unit is Hertz "Hz". Refer to the data sheet for the range of valid values. Not valid when WVTP is NOISE or DC.                    |
| PERI       | <period>    | := period. The unit is seconds "s". Refer to the data sheet for the range of valid values. Not valid when WVTP is NOISE or DC.                      |
| AMP        | <amplitude> | := amplitude. The unit is volts, peak-to-peak "Vpp". Refer to the data sheet for the range of valid values. Not valid when WVTP is NOISE or DC.     |
| OFST       | <offset>    | := offset. The unit is volts "V". Refer to the data sheet for the range of valid values. Not valid when WVTP is NOISE.                              |
| SYM        | <symmetry>  | := {0 to 100}. Symmetry of RAMP. The unit is "%". Only settable when WVTP is RAMP.  |
| DUTY       | <duty>      | := {0 to 100}. Duty cycle. The unit is "%". Value depends on frequency. Only settable when WVTP is SQUARE or PULSE.                                 |
| STDEV      | <stdev>     | := standard deviation of NOISE. The unit is volts "V". Refer to the data sheet for the range of valid values. Only settable when WVTP is NOISE.     |
| MEAN       | <mean>      | := mean of NOISE. The unit is volts "V". Refer to the data sheet for the range of valid values. Only settable when WVTP is NOISE.                   |
| WIDTH      | <width>     | := positive pulse width. The unit is seconds "s". Refer to the data sheet for the range of valid values. Only settable when WVTP is PULSE.          |



**QUERY SYNTAX**

<channel>: BaSic\_WaVe?

<channel>:= {C1}

**RESPONSE FORMAT**

<channel>:BSWV <parameter>

<parameter>:= All the parameters of the current basic waveform.

**EXAMPLE**

Change the waveform type of C1 to Ramp:

*C1:BSWV WVTP,RAMP*

Change the frequency of C1 to 2000 Hz:

*C1:BSWV FRQ,2000*

Set the amplitude of C1 to 3 Vpp:

*C1:BSWV AMP,3*

Return parameters of C1 from the device:

*C1:BSWV?*

Return:

*C1:BSWV*

*WVTP,SINE,FRQ,100HZ,PERI,0.01S,AMP,2V,OFST,0V,HLEV,1V,LLEV,-1V,PHSE,0*

**OUTPut****Command/Query****DESCRIPTION**

This command enables or disables the output port(s) at the front panel. The query returns “ON” or “OFF” and “LOAD”, “PLRT”, “RATIO” parameters.

**COMMAND SYNTAX**

<channel>:OUTPut <state>,LOAD,<load>

<channel>:= {C1}, SAG and the built-in waveform generator only support one output channel.

<state>:= {ON|OFF}

<load>:= {50|HZ}. The unit is ohm.

**QUERY SYNTAX**

<channel>:OUTPut?

**RESPONSE FORMAT**

<channel>:OUTP <state>,LOAD,<load>,PLRT,<polarity>

<state>:= {ON|OFF}

<load>:= {50|HZ}

<polarity>:= {NOR|INVT}, in which NOR refers to normal, and INVT refers to invert. SAG and the built-in waveform generator only support to set to NOR.

**EXAMPLE**

Turn on CH1:

*C1:OUTP ON*

Read CH1 output state:

*C1:OUTP?*

Return:

*C1:OUTP ON,LOAD,HZ,PLRT,NOR*

Set the load of CH1 to 50 ohm:

*C1:OUTP LOAD,50*

Set the load of CH1 to HiZ:

*C1:OUTP LOAD,HZ*

**SToreList****Query****DESCRIPTION**

This query is used to read the stored waveforms list with indexes and names. If the store unit is empty, the command will return "EMPTY" string.

**QUERY SYNTAX**

SToreList? [<location>]

<location>:= {BUILDDIN|USER}

**EXAMPLE**

Read all arbitrary data saved in the built-in waveform generator in SDS2000X Plus.

*STL?*

Return:

*STL M10, ExpFal, M100, ECG14, M101, ECG15, M102, LFPulse, M103, Tens1, M104, Tens2, M105, Tens3, M106, Airy, M107, Besselj, M108, Bessely, M109, Dirichlet, M11, ExpRise, M110, Erf, M111, Erfc, M112, ErfcInv, M113, ErfInv, M114, Laguerre, M115, Legend, M116, Versiera, M117, Weibull, M118, LogNormal, M119, Laplace, M12, LogFall, M120, Maxwell, M121, Rayleigh, M122, Cauchy, M123, CosH, M124, CosInt, M125, CotH, M126, CscH, M127, SecH, M128, SinH, M129, SinInt, M13, LogRise, M130, TanH, M131, ACosH, M132, ASecH, M133, ASinH, M134, ATanH, M135, ACsch, M136, ACoth, M137, Bartlett, M138, BohmanWin, M139, ChebWin, M14, Sqrt, M140, FlattopWin, M141, ParzenWin, M142, TaylorWin, M143, TukeyWin, M144, Duty01, M145, Duty02, M146, Duty04, M147, Duty06, M148, Duty08, M149, Duty10, M15, Root3, M150, Duty12, M151, Duty14, M152, Duty16, M153, Duty18, M154, Duty20, M155, Duty22, M156, Duty24, M157, Duty26, M158, Duty28, M159, Duty30, M16, X^2, M160, Duty32, M161, Duty34, M162, Duty36, M163, Duty38, M164, Duty40, M165, Duty42, M166, Duty44, M167, Duty46, M168, Duty48, M169, Duty50, M17, X^3, M170, Duty52, M171, Duty54, M172, Duty56, M173, Duty58, M174, Duty60, M175, Duty62, M176, Duty64, M177, Duty66, M178, Duty68, M179, Duty70, M18, Sinc, M180, Duty72, M181, Duty74, M182, Duty76, M183, Duty78, M184, Duty80, M185, Duty82, M186, Duty84, M187, Duty86, M188, Duty88, M189, Duty90, M19, Gaussian, M190, Duty92, M191, Duty94, M192, Duty96, M193, Duty98, M194, Duty99, M195, demo1\_375, M196, demo1\_16k, M197, demo2\_3k, M198, demo2\_16k, M2, StairUp, M20, Dlorentz, M21, Haversine, M22, Lorentz, M23, Gauspuls, M24, Gmonopuls, M25, Tripuls, M26, Cardiac, M27, Quake, M28, Chirp, M29, Twotone, M3, StairDn, M30, SNR, M31, Hamming, M32, Hanning, M33, kaiser, M34, Blackman, M35, Gausswin, M36, Triangle, M37, BlackmanH, M38, Bartlett-Hann, M39, Tan, M4, StairUD, M40, Cot, M41, Sec, M42, Csc, M43, Asin, M44, Acos, M45, Atan, M46, Acot, M47, Square, M48, SineTra, M49, SineVer, M5, Ppulse, M50, AmpALT, M51, AttALT, M52, RoundHalf, M53, RoundsPM, M54, BlaseiWave, M55, DampedOsc, M56, SwingOsc, M57, Discharge, M58, Pahcur, M59, Combin, M6, Npulse, M60, SCR, M61, Butterworth, M62, Chebyshev1, M63, Chebyshev2, M64, TV, M65, Voice, M66, Surge, M67,*

Radar, M68, Ripple, M69, Gamma, M7, Trapezia, M70, StepResp, M71, BandLimited, M72, CPulse, M73, CWPulse, M74, GateVibr, M75, LFMPulse, M76, MCNoise, M77, AM, M78, FM, M79, PFM, M8, Upramp, M80, PM, M81, PWM, M82, EOG, M83, EEG, M84, EMG, M85, Pulseilogram, M86, ResSpeed, M87, ECG1, M88, ECG2, M89, ECG3, M9, Dnramp, M90, ECG4, M91, ECG5, M92, ECG6, M93, ECG7, M94, ECG8, M95, ECG9, M96, ECG10, M97, ECG11, M98, ECG12, M99, ECG13

Read built-in wave data from a SDS2000X Plus built-in waveform generator:

STL? BUILDIN

Return:

STL M10, ExpFal, M100, ECG14, M101, ECG15, M102, LFPulse, M103, Tens1, M104, Tens2, M105, Tens3, M106, Airy, M107, Besselj, M108, Bessely, M109, Dirichlet, M11, ExpRise, M110, Erf, M111, Erfc, M112, ErfcInv, M113, ErfInv, M114, Laguerre, M115, Legend, M116, Versiera, M117, Weibull, M118, LogNormal, M119, Laplace, M12, LogFall, M120, Maxwell, M121, Rayleigh, M122, Cauchy, M123, CosH, M124, CosInt, M125, CotH, M126, CscH, M127, SecH, M128, SinH, M129, SinInt, M13, LogRise, M130, TanH, M131, ACosH, M132, ASecH, M133, ASinH, M134, ATanH, M135, ACsch, M136, ACoth, M137, Bartlett, M138, BohmanWin, M139, ChebWin, M14, Sqrt, M140, FlattopWin, M141, ParzenWin, M142, TaylorWin, M143, TukeyWin, M144, Duty01, M145, Duty02, M146, Duty04, M147, Duty06, M148, Duty08, M149, Duty10, M15, Root3, M150, Duty12, M151, Duty14, M152, Duty16, M153, Duty18, M154, Duty20, M155, Duty22, M156, Duty24, M157, Duty26, M158, Duty28, M159, Duty30, M16, X<sup>2</sup>, M160, Duty32, M161, Duty34, M162, Duty36, M163, Duty38, M164, Duty40, M165, Duty42, M166, Duty44, M167, Duty46, M168, Duty48, M169, Duty50, M17, X<sup>3</sup>, M170, Duty52, M171, Duty54, M172, Duty56, M173, Duty58, M174, Duty60, M175, Duty62, M176, Duty64, M177, Duty66, M178, Duty68, M179, Duty70, M18, Sinc, M180, Duty72, M181, Duty74, M182, Duty76, M183, Duty78, M184, Duty80, M185, Duty82, M186, Duty84, M187, Duty86, M188, Duty88, M189, Duty90, M19, Gaussian, M190, Duty92, M191, Duty94, M192, Duty96, M193, Duty98, M194, Duty99, M195, demo1\_375, M196, demo1\_16k, M197, demo2\_3k, M198, demo2\_16k, M2, StairUp, M20, Dlorentz, M21, Haversine, M22, Lorentz, M23, Gauspuls, M24, Gmonopuls, M25, Tripuls, M26, Cardiac, M27, Quake, M28, Chirp, M29, Twotone, M3, StairDn, M30, SNR, M31, Hamming, M32, Hanning, M33, kaiser, M34, Blackman, M35, Gausswin, M36, Triangle, M37, BlackmanH, M38, Bartlett-Hann, M39, Tan, M4, StairUD, M40, Cot, M41, Sec, M42, Csc, M43, Asin, M44, Acos, M45, Atan, M46, Acot, M47, Square, M48, SineTra, M49, SineVer, M5, Ppulse, M50, AmpALT, M51, AttALT, M52, RoundHalf, M53, RoundsPM, M54, BlaseiWave, M55, DampedOsc, M56, SwingOsc, M57, Discharge, M58, Pahcur, M59, Combin, M6, Npulse, M60, SCR, M61, Butterworth, M62, Chebyshev1, M63, Chebyshev2, M64, TV, M65, Voice, M66, Surge, M67, Radar, M68, Ripple, M69, Gamma, M7, Trapezia, M70, StepResp, M71, BandLimited, M72, CPulse, M73, CWPulse,

*M74, GateVibr, M75, LFMPulse, M76, MCNoise, M77, AM, M78, FM, M79, PFM, M8, Upramp, M80, PM, M81, PWM, M82, EOG, M83, EEG, M84, EMG, M85, Pulseilogram, M86, ResSpeed, M87, ECG1, M88, ECG2, M89, ECG3, M9, Dnramp, M90, ECG4, M91, ECG5, M92, ECG6, M93, ECG7, M94, ECG8, M95, ECG9, M96, ECG10, M97, ECG11, M98, ECG12, M99, ECG13*

## SYNC

### Command/Query

#### DESCRIPTION

This command sets or gets the synchronization signal.

#### COMMAND SYNTAX

<channel>:SYNC <state>

<channel>:= {C1}, SAG and the built-in waveform generator only support one output channel.

<state>:= {ON|OFF}

#### QUERY SYNTAX

<channel>:SYNC?

<channel>:= {C1}

#### RESPONSE FORMAT

<channel>:SYNC <state>,TYPE,<TYPE>

<channel>:= {C1}

<state>:= {ON|OFF}

<TYPE>:={CH1}, SAG and the built-in waveform generator only support one output channel, so it can only be CH1.

#### EXAMPLE

Turn on sync output:

*C1:SYNC ON*

Read state of CH1 sync.

*C1:SYNC?*

Return:

*C1:SYNC ON,TYPE,CH1*

## VOLTPRT

### Command/Query

#### DESCRIPTION

This command sets or gets the state of over-voltage protection.

#### COMMAND SYNTAX

VOLTPRT <state>

<state>:= {ON|OFF}

#### QUERY SYNTAX

VOLTPRT?

#### RESPONSE FORMAT

VOLTPRT <state>

## METEr Commands

The meter system commands are only for the multimeter functions of SHS800X/SHS1000X handheld digital oscilloscope. Support for configuration and measurement.

- ◆ **MMETer**
- ◆ **READ**
- ◆ **CONFigure Commands**
- ◆ **MEASure Commands**
- ◆ **SENSe Commands**

**MMETer****Command**

|                       |  |
|-----------------------|--|
| <b>DESCRIPTION</b>    | Enter the multimeter.  |
| <b>COMMAND SYNTAX</b> | MMETer <switch><br><switch>:= {ON OFF}                           |
| <b>EXAMPLE</b>        | Enter the multimeter<br><br>Command message:<br><i>MMETer ON</i> |

**READ****Query**

|                        |  |
|------------------------|--|
| <b>DESCRIPTION</b>     | Read measurement results.  |
| <b>QUERY SYNTAX</b>    | READ?  |
| <b>RESPONSE FORMAT</b> | MM_VALUE <value>   |
| <b>EXAMPLE</b>         | Read measurement results<br><br>Command message:<br><i>READ?</i><br><br>Response message:<br><i>MM_VALUE 0.00V</i> |



## **CONFigure Commands**

The CONFigure commands are the most concise way to configure measurements. These commands use default measurement configuration values. However, these commands do not automatically start measurements, so you can modify measurement attributes before initiating the measurement.

- ◆ **CONFigure**
- ◆ **CONFigure:CAPacitance**
- ◆ **CONFigure:CONTInuity**
- ◆ **CONFigure:CURREnt:AC**
- ◆ **CONFigure:CURREnt:DC**
- ◆ **CONFigure:DIODE**
- ◆ **CONFigure:RESistance**
- ◆ **CONFigure[:VOLTage]:AC**
- ◆ **CONFigure[:VOLTage]:DC**

**CONFigure****Query****DESCRIPTION**

Returns the present function and measured value. The present function name is returned in short format, such as ACV.

**QUERY SYNTAX**

CONFigure?

**RESPONSE FORMAT**

<func>

<func>:= {DCV|ACV|RES|DIODE|CONTINUITY|CAP|DCI|ACI}

**EXAMPLE**

Return the present function and measured value:

Command message:

*CONFigure?*

Response message:

*DCV -04.mV*

**CONFigure:CONTInuity****Command****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for continuity measurements.

The READ? and MEASure:CONTInuity? queries return the measured resistance. If the resistance is greater than 600Ω, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**COMMAND SYNTAX**

CONFigure:CONTInuity

**EXAMPLE**

Configure the instrument for continuity measurements, and read the measurement:

Command message:

*CONF:CONT  
READ?*

Response message:

*Overload*

**CONFigure:CURRent:AC****Command/Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for AC current measurements. Also specifies the range through the incoming parameters.

- You can let autoranging select the measurement range, or you can manually select a fixed range. Autoranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (autoranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

**COMMAND SYNTAX**

CONFigure:CURRent:AC <range>

<range>:= {60mA|600mA|6A|10A|AUTO|MIN|MAX|DEF}

Default: AUTO

**EXAMPLE**

Configure AC current measurements using the 6A range. And read measurement:

Command message:  
*CONF:CURR:AC 6A*  
*READ?*

Response message:  
*+4.32133675E-04*

**CONFigure:CURRent:DC****Command/Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for DC current measurements. Also specifies the range through the incoming parameters.

**COMMAND SYNTAX**

CONFigure:CURRent:DC <range>

<range>:= {60mA|600mA|6A|10A|AUTO|MIN|MAX|DEF}  
Default: AUTO

**EXAMPLE**

Configure DC current measurements using the 6A range. And read measurement:

Command message:  
*CONF:CURR:DC 6A*  
*READ?*

Response message:  
*+4.32133675E-04*

**CONFigure:DIODe****Command****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for diode measurements.

- The range is fixed for diode tests is 2 VDC.
- The READ? and MEASure:DIODe? queries return the measured voltage. If the voltage is greater than 2V, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

**COMMAND SYNTAX**

CONFigure:DIODe

**EXAMPLE**

Configure diode measurement ,and read the measurement:

Command message:  
*CONF:DIOD*  
*READ?*

Response message:  
*Overload*

**CONFigure:RESistance****Command****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for 2-wire (RESistance) resistance measurements. Also specifies the range and resolution.

- You can let autoranging select the measurement range, or you can manually select a fixed range. Autoranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (autoranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word Overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

**COMMAND SYNTAX**

CONFigure:RESistance <range>

<range>:= {600|6k|60k|600k|6M|60M|AUTO|MIN|MAX|DEF}

Default: AUTO

**EXAMPLE**

Configure 2-wire resistance measurements using the 600  $\Omega$  range. Make and read measurements.

Command message:

*CONF:RES 600*  
*READ?*

Response message:

*+6.71881065E+01*

## CONFigure[:VOLTage]:AC

### Command

#### DESCRIPTION

Sets all measurement parameters and trigger parameters to their default values for AC voltage measurements. Also specifies the range through the incoming parameters.

- You can let autoranging select the measurement range, or you can manually select a fixed range. Autoranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (autoranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word Overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

#### COMMAND SYNTAX

CONFigure[:VOLTage]:AC <range>

| Model    | <range>  |
|----------|--|
| SHS800X  | {60mV 600mV 6V 60V 600V AUTO MIN MAX DEF}      |
| SHS1000X | {60mV 600mV 6V 60V 600V 750V AUTO MIN MAX DEF} |

Default: AUTO

#### EXAMPLE

Configure AC voltage measurements using the 60 V range.  
Read measurements:

Command message:

```
CONF:VOLT:AC 60
READ?
```

Response message:

```
+2.43186951E-02
```

## CONFigure[:VOLTage]:DC

### Command

#### DESCRIPTION

Sets all measurement parameters and trigger parameters to their default values for DC voltage measurements. Also specifies the range through the incoming parameters.

- You can let autoranging select the measurement range, or you can manually select a fixed range. Autoranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (autoranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word Overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

#### COMMAND SYNTAX

CONFigure[:VOLTage]:DC <range>

| Model    | <range>   |
|----------|---|
| SHS800X  | {60mV 600mV 6V 60V 600V AUTO MIN MAX DEF}       |
| SHS1000X | {60mV 600mV 6V 60V 600V 1000V AUTO MIN MAX DEF} |

Default: AUTO

#### EXAMPLE

Configure DC voltage measurements using the 60 V range.  
Read measurements:

Command message:

```
CONF:VOLT:DC 60
READ?
```

Response message:

```
+2.43186951E-02
```

**CONFigure:CAPacitance****Command****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for capacitance measurement.

- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word Overload on front panel and returns "Overload" from the remote interface.
- Use READ? to start the measurement.

**COMMAND SYNTAX**

CONFigure:CAPacitance

**EXAMPLE**

Configure capacitance measurement using the 4uF range.  
Read measurements:

Command message:

```
CONF:CAP  
READ?
```

Response message:

```
+7.26141264E-10
```



## MEASure Commands

The MEASure queries are the easiest way to program measurements because they always use default measurement parameters. You set the function, range in one command, but you cannot change other parameters from their default values. The results are sent directly to the instrument's output buffer.

**Note:** A MEASure query is functionally equivalent to sending CONFigure followed immediately by READ? The difference is that CONFigure commands allow you to change parameters between the CONFigure and the READ?

- ◆ **MEASure:CONTInuity**
- ◆ **MEASure:CURRent:AC**
- ◆ **MEASure:CURRent:DC**
- ◆ **MEASure:DIODE**
- ◆ **MEASure:RESistance**
- ◆ **MEASure[:VOLTage]:AC**
- ◆ **MEASure[:VOLTage]:DC**
- ◆ **MEASure[:VOLTage]:AC**
- ◆ **MEASure:CAPacitance**

**MEASure:CONTInuity****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for continuity test and immediately triggers a measurement. The results are sent directly to the instrument's output buffer.

- The range is fixed at 600Ω for continuity tests (a 2-wire resistance measurement).
- The instrument beeps (if the beeper is enabled) for each measurement less than or equal to the continuity threshold, and the actual resistance measurement appears on the display.
- The READ? and MEASure:CONTInuity? queries return the measured resistance. If the resistance is greater than 600Ω, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:CONTInuity?

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure the instrument for continuity measurements. Then Read measurements:

Command message:

*MEAS:CONT?*

Response message:

*+9.84739065E+02*

**MEASure:CURRent:AC****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for AC current measurements and immediately triggers a measurement. Also specifies the range through the incoming parameters.

- You can let auto ranging select the measurement range, or you can manually select a fixed range. Auto ranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (auto ranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word Overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:CURRent:AC? <range>

<range>:={60mA|600mA|6A|10A|AUTO}  
Default: AUTO

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure AC current measurement using the 6A range. Read measurements:

Command message:  
*MEAS:CURR:AC? 6*

Response message:  
*+4.32133675E-04*

**MEASure:CURRent:DC****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for DC current measurements and immediately triggers a measurement. Also specifies the range through the incoming parameters.

**QUERY SYNTAX**

MEASure:CURRent:DC? <range>

<range>:={60mA|600mA|6A|10A|AUTO}  
Default: AUTO

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure DC current measurement using the 6A range. Read measurements:

Command message:  
*MEAS:CURR:DC? 6*

Response message:  
*+4.32133675E-04*

**MEASure:DIODe****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values for diode test measurements and immediately triggers a measurement. The results are sent directly to the instrument's output buffer.

- The range and resolution are fixed for diode tests: the range is 2 VDC.
- The READ? and MEASure:DIODe? queries return the measured voltage. If the voltage is greater than 2V, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:DIODe?

**RESPONSE FORMAT**

&lt;value&gt;

**EXAMPLE**

Configure and read a default diode measurement:

*MEAS:DIOD?*

Response message:

*+9.84733701E-01*

**MEASure:RESistance****Query****DESCRIPTION**

Sets all measurement to their default values for 2-wire (RESistance) measurements, and immediately triggers a measurement. The results are sent directly to the instrument's output buffer. Also specifies the range through the incoming parameters.

- You can let auto ranging select the measurement range, or you can manually select a fixed range. Auto ranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (auto ranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:RESistance? <range>

<range>:={600|6k|60k|600k|6M|60M}

Default: AUTO

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure 2-wire resistance measurements using the 600Ω range. Make and read measurements.

Command message:

*MEAS:RES? 600*

Response message:

*+6.71881065E+01*

**MEASure[:VOLTage]:AC****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values and immediately triggers a measurement. The results are sent directly to the instrument's output buffer. Also specifies the range through the incoming parameters.

- You can let auto ranging select the measurement range, or you can manually select a fixed range. Auto ranging conveniently selects the range for each measurement based on the input signal. For fastest measurements, use manual ranging (auto ranging may require additional time for range selection).
- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:VOLTage:AC? <range>

| Model    | <range>                                     |
|----------|---|
| SHS800X  | {60mV 600mV 6V 60V 600V }                   |
| SHS1000X | {60mV 600mV 6V 60V 600V 1000V(DC)/750V(AC)} |

Default: AUTO

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure AC voltage measurements using the 600 V range. Read measurements:

Command message:  
*MEAS:VOLT:AC? 600*

Response message:  
*+2.43186951E-02*

**MEASure[:VOLTage]:DC****Query****DESCRIPTION**

Sets all measurement parameters and trigger parameters to their default values and immediately triggers a measurement. The results are sent directly to the instrument's output buffer. Also specifies the range through the incoming parameters.

**QUERY SYNTAX**

MEASure:VOLTage:DC? <range>

| Model    | <range>                                     |
|----------|---|
| SHS800X  | {60mV 600mV 6V 60V 600V }                   |
| SHS1000X | {60mV 600mV 6V 60V 600V 1000V(DC)/750V(DC)} |

Default: AUTO

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Configure DC voltage measurements using the 600 V range. Read measurements:

Command message:  
*MEAS:VOLT:DC? 600*

Response message:  
*+2.43186951E-02*



**MEASure:CAPacitance****Query****DESCRIPTION**

Sets all measurement parameters to their default values for capacitance measurement.

- If the input signal is greater than can be measured on the specified manual range, the instrument displays the word overload on front panel and returns "Overload" from the remote interface.

**QUERY SYNTAX**

MEASure:CAPacitance?

**RESPONSE FORMAT**

<value>

**EXAMPLE**

Command message:  
*MEAS:CAP?*

Response message:  
*+7.26141264E-10*

**SENSe Commands**

- ◆ [SENSe:]CAPacitance:NULL
- ◆ [SENSe:]CURRent:AC:NULL
- ◆ [SENSe:]CURRent:AC:SELEct
- ◆ [SENSe:]CURRent:DC:NULL
- ◆ [SENSe:]CURRent:DC:SELEct
- ◆ [SENSe:]RESistance:NULL
- ◆ [SENSe:]VOLTagE:AC:NULL
- ◆ [SENSe:]VOLTagE:AC:SELEct
- ◆ [SENSe:]VOLTagE:DC:NULL
- ◆ [SENSe:]VOLTagE:DC:SELEct

**[SENSe:]CURRent:AC:NULL****Command****DESCRIPTION**

Enables or disables the relative function for AC current measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

- Enabling the scaling function also enables automatic relative value selection ([SENSe:]CURRent:AC:NULL ON).
- The instrument disables the relative function after a Factory Reset or CONFigure function.

**COMMAND SYNTAX**

[SENSe:]CURRent:AC:NULL <state>

<state>:={ON|OFF}

Default: OFF

**EXAMPLE**

Configure AC current measurements.

```
CONF:CURR:AC  
CURR:AC:NULL ON  
READ?
```

Response message:

```
MM_VALUE 0.00V
```

**[SENSe:]CURRent:DC:NULL****Command****DESCRIPTION**

Enables or disables the relative function for DC current measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

- Enabling the scaling function also enables automatic relative value selection ([SENSe:]CURRent:DC:NULL ON).
- The instrument disables the relative function after a Factory Reset or CONFigure function.

**COMMAND SYNTAX**

[SENSe:]CURRent:DC:NULL <state>

<state>:={ON|OFF}

Default: OFF

**EXAMPLE**

Configure DC current measurements.

```
CONF:CURR:DC
CURR:DC:NULL ON
READ?
```

Response message:

```
MM_VALUE 0.00V
```

**[SENSe:]CURRent:AC:SELEct****Command****DESCRIPTION**

mA or A selection for AC current measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

**COMMAND SYNTAX**

[SENSe:]CURRent:AC:SELEct <unit>

<unit>:={MA|A}

**EXAMPLE**

```
CONF:CURR:AC
CURR:AC:SELE MA
```

**[SENSe:]CURRent:DC:SELEct****Command****DESCRIPTION**

mA or A selection for DC current measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

**COMMAND SYNTAX**

[SENSe:]CURRent:DC:SELEct <unit>

<unit>:={MA|A}

**EXAMPLE**

```
CONF:CURR:DC
CURR:DC:SELE MA
```

**[SENSe:]RESistance:NULL****Command****DESCRIPTION**

Enables or disables the relative function for resistance measurements.

- Enabling the scaling function also enables automatic relative value selection ([SENSe:]RESistance:NULL ON).
- The instrument disables the relative function after a Factory Reset or CONFigure function.

**COMMAND SYNTAX**

[SENSe:]RESistance:NULL <state>

<state>:={ON|OFF}

Default: OFF

**EXAMPLE**

Configure 2-wire resistance measurements, provide 1.5K $\Omega$  measurement resistance. Make and read measurements

Command message:

```
CONF:RES
RES:NULL ON
READ?
```

Response message:

```
0
```

**[SENSe:]VOLTage:AC:NULL****Command****DESCRIPTION**

Enables or disables the relative function for AC voltage measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

The instrument disables the relative function after a Factory Reset or CONFIGure function.

**COMMAND SYNTAX**

[SENSe:]VOLTage:AC:NULL <state>

<state>:={ON|OFF}

Default:OFF

**EXAMPLE**

Configure AC voltage measurements, Provide 1.5V AC voltage signal, Make and read measurements:

Command message:

```
CONF:VOLT:AC  
VOLT:AC:NULL ON  
READ?
```

Response message:

```
MM_VALUE 00.04V
```

**[SENSe:]VOLTage:DC:NULL****Command****DESCRIPTION**

Enables or disables the relative function for DC voltage measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

The instrument disables the relative function after a Factory Reset or CONFigure function.

**COMMAND SYNTAX**

[SENSe:]VOLTage:DC:NULL <state>

<state>:={ON|OFF}

Default: OFF

**EXAMPLE**

Configure DC voltage measurements, Provide 1.5V DC voltage signal, Make and read measurements:

Command message:

```
CONF:VOLT:DC  
VOLT:DC:NULL ON  
READ?
```

Response message:

```
MM_VALUE 00.04V
```

**[SENSe:]VOLTage:AC:SELEct****Commend****DESCRIPTION**

mV or V selection for AC voltage measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

**COMMAND SYNTAX**

[SENSe:]VOLTage:AC:SELEct <unit>

<unit>:={MV|V}

**EXAMPLE**

```
CONF:VOLT:AC  
VOLT:AC:SELE V
```

**[SENSe:]VOLTage:DC:SELEct****Commend****DESCRIPTION**

mV or V selection for DC voltage measurements.

**Note:** This parameter is not shared between AC and DC measurements. The parameters are independent for AC and DC measurements.

**COMMAND SYNTAX**

[SENSe:]VOLTage:DC:SELEct <unit>

<unit>:={MV|V}

**EXAMPLE**

```
CONF:VOLT:DC  
VOLT:DC:SELE V
```



## [SENSe:]CAPacitance:NULL

### Commend

#### DESCRIPTION

Enable or disable the relative function for capacitance measurement.

#### COMMAND SYNTAX

[SENSe:]CAPacitance:NULL <state>

<state>:={ON|OFF}

Default: OFF

#### EXAMPLE

Configure capacitance measurements, make and read measurements:

Command message:

```
CONF:CAP  
CAP:NULL ON  
READ?
```

Response message:

```
MM_VALUE 0.00nF
```

## Programming Examples

This chapter gives some examples for the programmer. In these examples you can see how to use VISA or sockets, in combination with the commands described above to control the oscilloscope. By following these examples, you can develop many more applications.

### ◆ VISA Examples

- ◆ VC++ Example
- ◆ VB Example
- ◆ MATLAB Example
- ◆ LabVIEW Example
- ◆ C# Example

### ◆ Examples of Using Sockets

- ◆ Python Example
- ◆ C Example

### ◆ Common Command Examples

- ◆ Read Waveform Data Example
- ◆ Read Waveform Data of Digital Example
- ◆ Read Waveform Data of FFT Example
- ◆ Read Sequence Waveform Data Example
- ◆ Screen Dump (PRINT) Example

## VISA Examples

### VC++ Example

**Environment:** Win7 32-bit, Visual Studio.

**Description:** Use National Instruments VISA to control the device with USBTMC or TCP/IP access. Perform a write and read operation.

#### Steps:

1. Open Visual Studio, create a new VC++ win32 project.
2. Set the project environment to use the NI-VISA library. There are two ways to use NI-VISA, static or automatic:
  - a) Static:

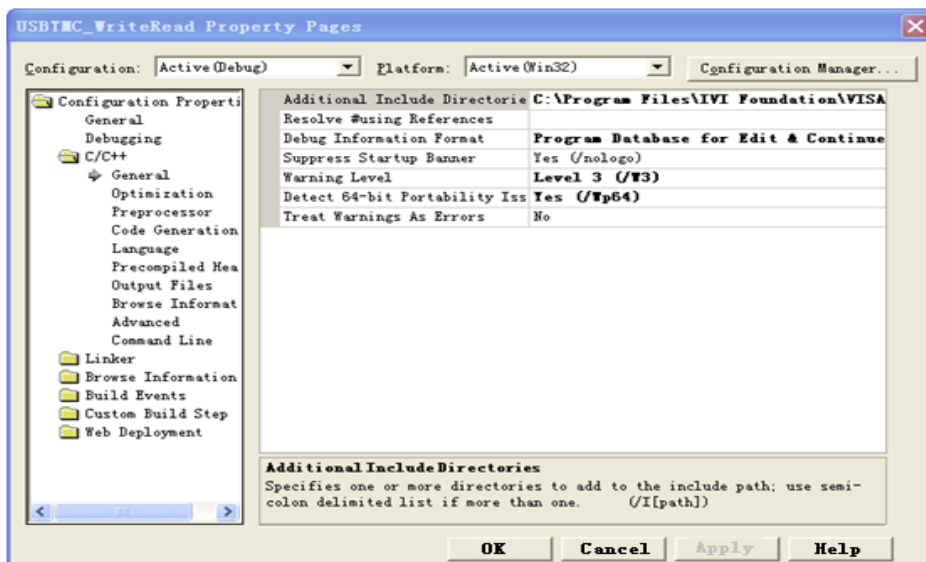
Find the files visa.h, visatype.h, visa32.lib in the NI-VISA installation path, copy them to your project, and add them into the project. In the projectname.cpp file, add the follow two lines:

```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

- b) Automatic:

Set the .h file include directory, the NI-VISA install path, in our computer we set the path is: C:\Program Files\IVI Foundation\IVISA\WinNT\include. Set this path to: project->properties->C/C++->General->Additional Include Directories.

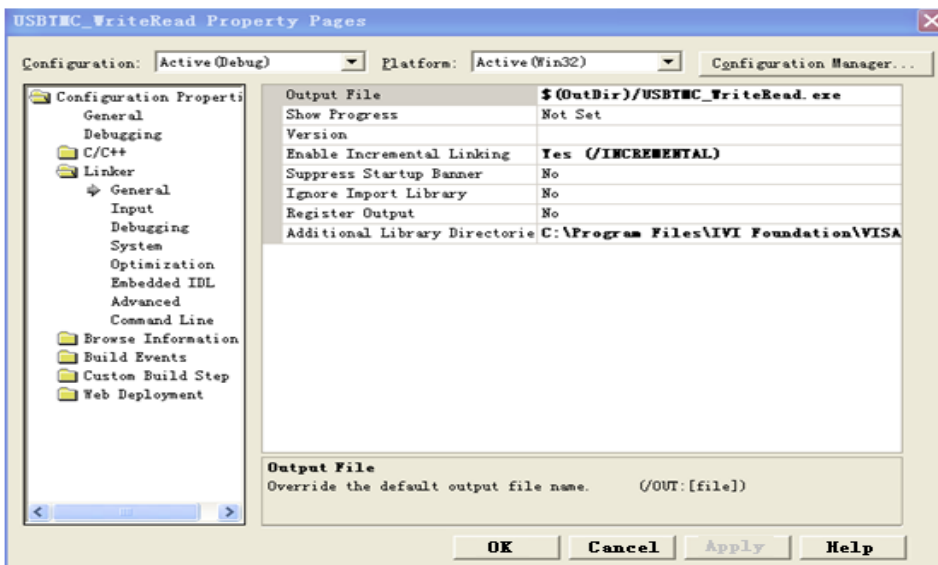
See the picture:



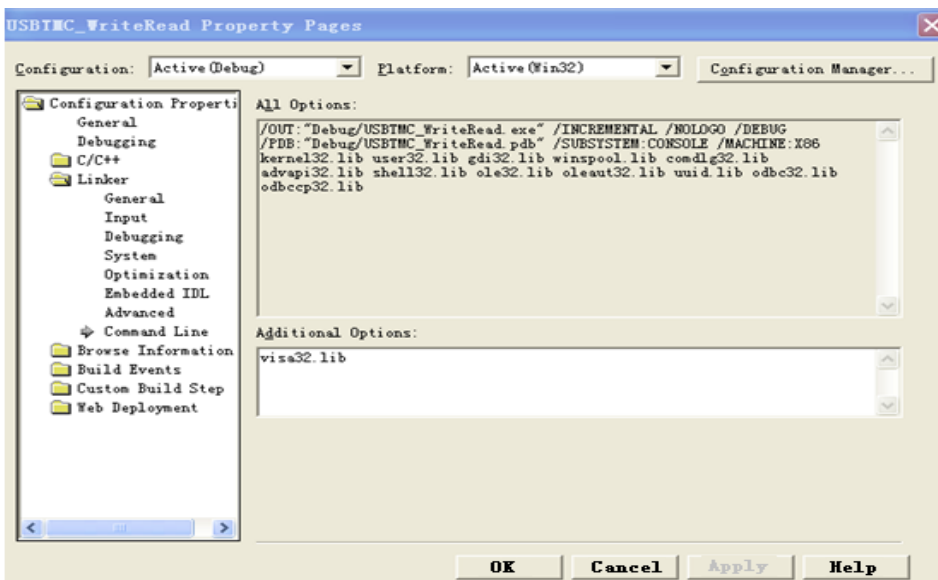
Set lib path set lib file:

Set lib path: the NI-VISA install path, in our computer we set the path is C:\Program Files\IVI Foundation\IVISA\WinNT\lib\msc. Set this path to: project->properties->Linker->General->Additional Library Directories.

As shown in the pictures below:



Set lib file:project->properties->Linker->Command Line->Additional Options: visa32.lib



Include visa.h file in the projectname.cpp file:

```
#include <visa.h>
```

3. Coding:
  - a) USBTMC:

```
Int Usbtmc_test()
{
    /* This code demonstrates sending synchronous read & write commands */
    /* to an USB Test & Measurement Class (USBTMC) instrument using */
}
```

```

/* NI-VISA          */
/* The example writes the "*IDN?\n" string to all the USBTMC */
/* devices connected to the system and attempts to read back */
/* results using the write and read functions.          */
/* The general flow of the code is */
/*  Open Resource Manager          */
/*  Open VISA Session to an Instrument          */
/*  Write the Identification Query Using viPrintf          */
/*  Try to Read a Response With viScanf          */
/*  Close the VISA Session          */
/*****/
ViSession defaultRM;
ViSession instr;
ViUInt32 numInstrs;
ViFindList findList;
ViUInt32 retCount;
ViUInt32 writeCount;
ViStatus status;
char instrResourceString[VI_FIND_BUFLEN];
unsigned charbuffer[100];
charstringinput[512];
int i;
/** First we must call viOpenDefaultRM to get the manager
 * handle. We will store this handle in defaultRM.*/
status= ViOpenDefaultRM (&defaultRM);
if (status<VI_SUCCESS)
{
    printf ("Could not open a session to the VISA Resource Manager!\n");
    return status;
}
/* Find all the USB TMC VISA resources in our system and store the number of resources
in the system in numInstrs.          */
status = viFindRsrc (defaultRM, "USB?*INSTR", &findList, &numInstrs,
instrResourceString);
if (status<VI_SUCCESS)
{
    printf ("An error occurred while finding resources.\nHit enter to continue.");
    fflush(stdin);
}

```

```

    getchar();
    viClose (defaultRM);
    return  status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
 * We must use the handle from viOpenDefaultRM and we must
 * also use a string that indicates which instrument to open. This
 * is called the instrument descriptor. The format for this string
 * can be found in the function panel by right clicking on the
 * descriptor parameter. After opening a session to the
 * device, we will get a handle to the instrument which we
 * will use in later VISA functions. The AccessMode and Timeout
 * parameters in this function are reserved for future
 * functionality. These two parameters are given the value VI_NULL.*/
for (i= 0; i<numInstrs; i++)
{
    if (i> 0)
    {
        viFindNext (findList, instrResourceString);
    }
    status = viOpen (defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
    if (status<VI_SUCCESS)
    {
        printf ("Cannot open a session to the device %d.\n", i+1);
        continue;
    }
    /** * At this point we now have a session open to the USB TMC instrument.
 * We will now use the viPrintf function to send the device the string "*IDN?\n",
 * asking for the device's identification. */
    char * cmmand ="*IDN?\n";
    status = viPrintf (instr, cmmand);
    if (status<VI_SUCCESS)
    {
        printf ("Error writing to the device %d.\n", i+1);
        status = viClose (instr);
        continue;
    }
    /** Now we will attempt to read back a response from the device to

```

```
    * the identification query that was sent. We will use the viScanf
    * function to acquire the data.
    * After the data has been read the response is displayed.*/
    status = viScanf(instr, "%t", buffer);
    if (status<VI_SUCCESS)
    {
        printf ("Error reading a response from the device %d.\n", i+1);
    }
    else
    {
        printf ("\nDevice %d: %s\n", i+1,retCount, buffer);
    }
    status = viClose (instr);
}
/* Now we will close the session to the instrument using
 * viClose. This operation frees all system resources.      */
status = viClose (defaultRM);
printf("Press 'Enter' to exit.");
fflush(stdin);
getchar();
return 0;
}
```

## b) TCP/IP:

```
int TCP_IP_Test(char *pIP)
{
    char outputBuffer[VI_FIND_BUFLEN];
    ViSession defaultRM, instr;
    ViStatus status;
    ViUInt32 count;
    ViUInt16 portNo;
    /* First we will need to open the default resource manager. */
    status = viOpenDefaultRM (&defaultRM);
    if (status<VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
    }
    /* Now we will open a session via TCP/IP device */
```

```
charhead[256] = "TCPIP0::";
chartail[] = "::INSTR";
charresource [256];
strcat(head,pIP);
strcat(head,tail);
status = viOpen (defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
if (status<VI_SUCCESS)
{
    printf ("An error occurred opening the session\n");
    viClose(defaultRM);
}
status = viPrintf(instr, "idn?\n");
status = viScanf(instr, "%t", outputBuffer);
if (status<VI_SUCCESS)
{
    printf("viRead failed with error code: %x \n",status);
    viClose(defaultRM);
}
else
{
    printf ("\ndata read from device: %*s\n", 0,outputBuffer);
}
status = viClose (instr);
status = viClose (defaultRM);
printf("Press 'Enter' to exit.");
fflush(stdin);
getchar();
return 0;
}
```



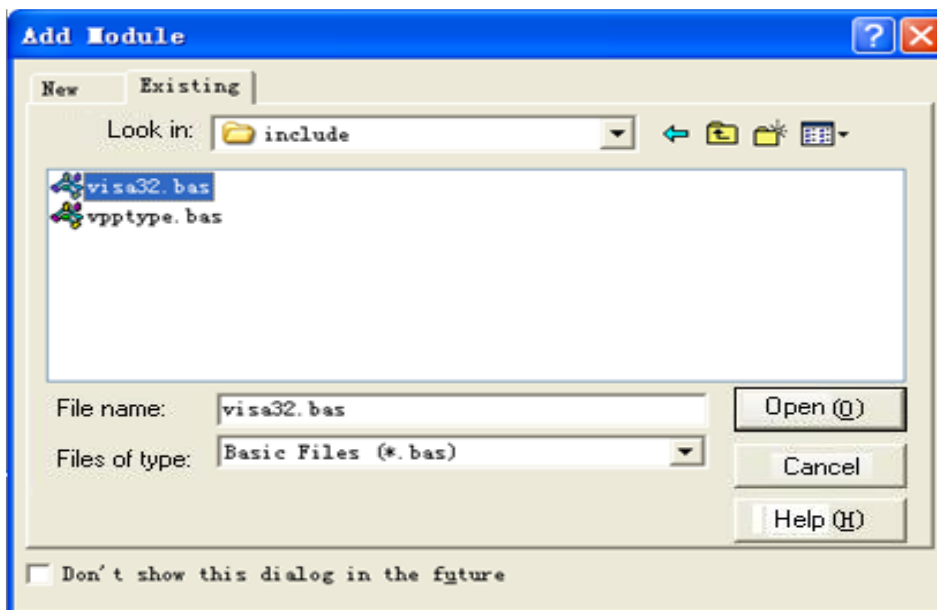
## VB Example

**Environment:** Windows7 32-bit, Microsoft Visual Basic 6.0

**Description:** The function of this example: Use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

**Steps:**

1. Open Visual Basic, and build a standard application program project.
2. Set the project environment to use the NI-VISA lib: Click the Existing tab of Project->Add Module, search the visa32.bas file in the "include" folder under the NI-VISA installation path and add the file, as shown in the figure below:



3. Coding:

- a) USBTMC:

`Private Function Usbtmc_test() As Long`

```
' This code demonstrates sending synchronous read & write commands
' to an USB Test & Measurement Class (USBTMC) instrument using
' NI-VISA
' The example writes the "*IDN?\n" string to all the USBTMC
' devices connected to the system and attempts to read back
' results using the write and read functions.
' The general flow of the code is
'   Open Resource Manager
'   Open VISA Session to an Instrument
'   Write the Identification Query Using viWrite
'   Try to Read a Response With viRead
```

```
' Close the VISA Session
Const MAX_CNT = 200

Dim defaultRM As Long
Dim instrsesn As Long
Dim numInstrs As Long
Dim findList As Long
Dim retCount As Long
Dim writeCount As Long
Dim status As Long
Dim instrResourceString As String * VI_FIND_BUFLen
Dim buffer As String * MAX_CNT
Dim i As Integer
' First we must call viOpenDefaultRM to get the manager
' handle. We will store this handle in defaultRM.
status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
    Debug.Print "Could not open a session to the VISA Resource Manager!"
    Usbtmc_test = status
    Exit Function
End If

' Find all the USB TMC VISA resources in our system and store the
' number of resources in the system in numInstrs.
status= ViFindRsrc(defaultRM,"USB?*INSTR",findList,numInstrs,instrResourceString)
If (status < VI_SUCCESS) Then
    Debug.Print "An error occurred while finding resources."
    viClose (defaultRM)
    Usbtmc_test = status
    Exit Function
End If

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
' also use a string that indicates which instrument to open. This
' is called the instrument descriptor. The format for this string
' can be found in the function panel by right clicking on the
' descriptor parameter. After opening a session to the
```

```
' device, we will get a handle to the instrument which we
' will use in later VISA functions. The AccessMode and Timeout
' parameters in this function are reserved for future
' functionality. These two parameters are given the value VI_NULL.
For i = 0 To numInstrs
  If (i > 0) Then
    status = viFindNext(findList, instrResourceString)
  End If
  status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
  If (status < VI_SUCCESS) Then
    Debug.Print "Cannot open a session to the device ", i + 1
    GoTo NextFind
  End If

  ' At this point we now have a session open to the USB TMC instrument.
  ' We will now use the viWrite function to send the device the string "*IDN?",
  ' asking for the device's identification.
  status = viWrite(instrsesn, "*IDN?", 5, retCount)
  If (status < VI_SUCCESS) Then
    Debug.Print "Error writing to the device."
    status = viClose(instrsesn)
    GoTo NextFind
  End If

  ' Now we will attempt to read back a response from the device to
  ' the identification query that was sent. We will use the viRead
  ' function to acquire the data.
  ' After the data has been read the response is displayed.
  status = viRead(instrsesn, buffer, MAX_CNT, retCount)
  If (status < VI_SUCCESS) Then
    Debug.Print "Error reading a response from the device.", i + 1
  Else
    Debug.Print i + 1, retCount, buffer
  End If
  status = viClose(instrsesn)
Next i

' Now we will close the session to the instrument using
```

```
' viClose. This operation frees all system resources.
```

```
status = viClose(defaultRM)
```

```
Usbtmc_test = 0
```

```
End Function
```

b) TCP/IP:

```
Private Function TCP_IP_Test(ip As String) As Long
```

```
Dim outputBuffer As String * VI_FIND_BUFLen
```

```
Dim defaultRM As Long
```

```
Dim instrsesn As Long
```

```
Dim status As Long
```

```
Dim count As Long
```

```
' First we will need to open the default resource manager.
```

```
status = viOpenDefaultRM (defaultRM)
```

```
If (status < VI_SUCCESS) Then
```

```
    Debug.Print "Could not open a session to the VISA Resource Manager!"
```

```
    TCP_IP_Test = status
```

```
    Exit Function
```

```
End If
```

```
' Now we will open a session via TCP/IP device
```

```
status = viOpen(defaultRM, "TCPIP0::" + ip + "::INSTR", VI_LOAD_CONFIG, VI_NULL,  
instrsesn)
```

```
If (status < VI_SUCCESS) Then
```

```
    Debug.Print "An error occurred opening the session"
```

```
    viClose (defaultRM)
```

```
    TCP_IP_Test = status
```

```
    Exit Function
```

```
End If
```

```
status = viWrite(instrsesn, "*IDN?", 5, count)
```

```
If (status < VI_SUCCESS) Then
```

```
    Debug.Print "Error writing to the device."
```

```
End If
```

```
status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLen, count)
```

```
If (status < VI_SUCCESS) Then
```

```
    Debug.Print "Error reading a response from the device.", i + 1
```

Else

    Debug.Print "read from device:", outputBuffer

End If

status = viClose(instrsesn)

status = viClose(defaultRM)

TCP\_IP\_Test = 0

End Function

## MATLAB Example

**Environment:** Windows7 32-bit, MATLAB R2010b

**Description:** The function of this example: Use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

**Steps:**

1. Open MATLAB, and modify the current directory. In this demo, the current directory is modified to D:\USBTMC\_TCPIP\_Demo.
2. Click File>>New>>Script in the Matlab interface to create an empty M file.
3. Coding:
  - a) USBTMC:

```
function USBTMC_test()
% This code demonstrates sending synchronous read & write commands
% to an USB Test & Measurement Class (USBTMC) instrument using
% NI-VISA

%Create a VISA-USB object connected to a USB instrument
vu = visa('ni','USB0::0xF4EC::0xEE38::0123456789::INSTR');

%Open the VISA object created
fopen(vu);

%Send the string "*IDN?",asking for the device's identification.
fprintf(vu,'*IDN?');

%Request the data
outputbuffer = fscanf(vu);
disp(outputbuffer);

%Close the VISA object
fclose(vu);
delete(vu);
clear vu;

end
```

- b) TCP/IP:

```
function TCP_IP_test( IPstr )
% This code demonstrates sending synchronous read & write commands
% to an TCP/IP instrument using NI-VISA

%Create a VISA-TCPIP object connected to an instrument
%configured with IP address.
vt = visa('ni',['TCPIP0::',IPstr, '::INSTR']);

%Open the VISA object created
fopen(vt);

%Send the string "*IDN?", asking for the device's identification.
fprintf(vt,'*IDN?');

%Request the data
outputbuffer = fscanf(vt);
disp(outputbuffer);

%Close the VISA object
fclose(vt);
delete(vt);
clear vt;

end
```

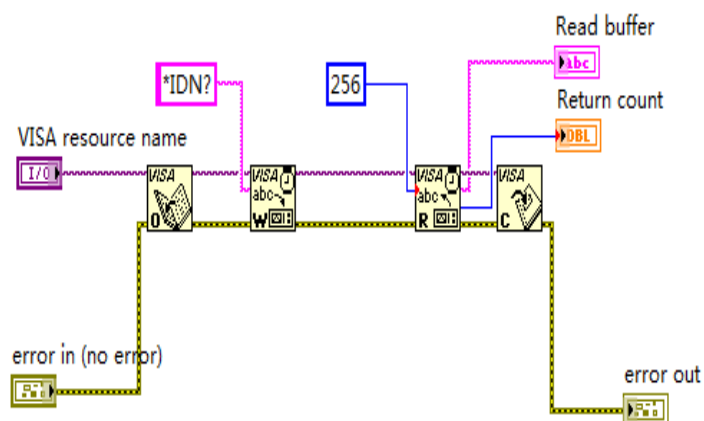
## LabVIEW Example

**Environment:** Windows7 32-bit, LabVIEW 2011

**Description:** The functions of this example: use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

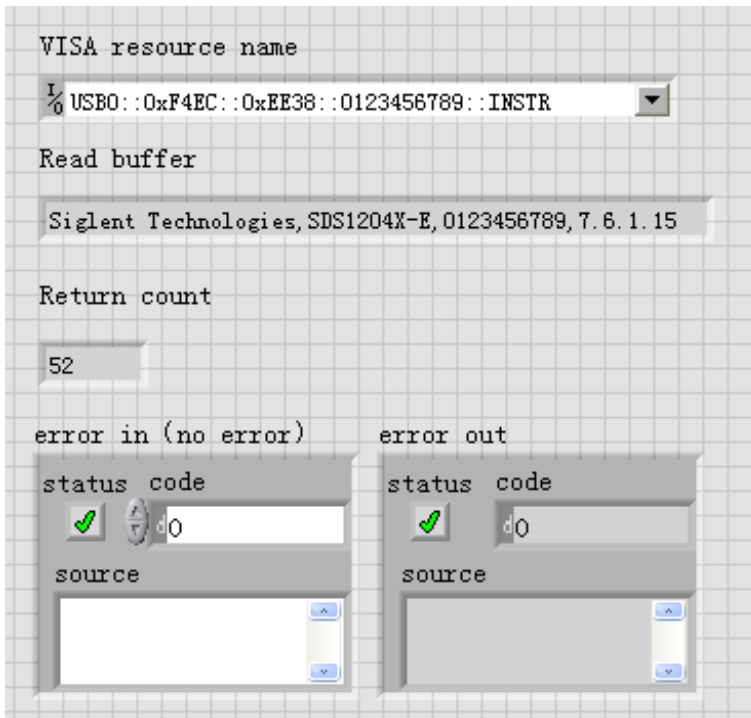
### Steps:

1. Open LabVIEW, create a VI file.
2. Add controls. Right-click in the **Front Panel** interface, select and add **VISA resource name**, error in, error out and some indicators from the Controls column.
3. Open the **Block Diagram** interface. Right-click on the **VISA resource name** and you can select and add the following functions from VISA Palette from the pop-up menu: **VISA Write**, **VISA Read**, **VISA Open** and **VISA Close**.
4. The connection is as shown in the figure below:



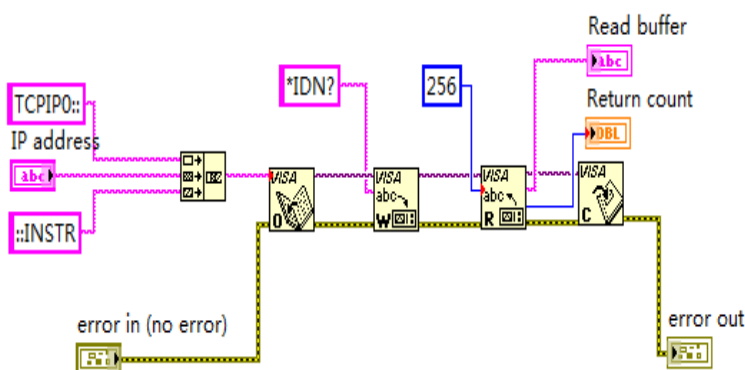
5. Select the device resource from the VISA Resource Name list box and run the program.





In this example, the VI opens a VISA session to a USBTMC device, writes a command to the device, and reads back the response. After all communication is complete, the VI closes the VISA session.

6. Communicating with the device via TCP/IP is similar to USBTMC. But you need to change VISA Write and VISA Read Function to Synchronous I/O. The LabVIEW default is asynchronous I/O. Right-click the node and select Synchronous I/O Mod>>Synchronous from the shortcut menu to write or read data synchronously.
7. The connection is as shown in the figure below:



8. Input the IP address and run the program.

IP address





10.11.25.232

Read buffer

Siglent Technologies, SDS1204X-E, 0123456789, 7.6.1.15

Return count

52

|   |   |
|---|---|
| error in (no error)   | error out   |
| <p>status code</p> <p>  0</p> <p>source</p> <p><input type="text"/></p> | <p>status code</p> <p>  0</p> <p>source</p> <p><input type="text"/></p> |

## C# Example

**Environment:** Windows7 32-bit, Visual Studio 2008/2010

**Description:** The functions of this example: use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

**Steps:**

1. Open Visual Studio, create a new C# project.
2. Cut-and-paste the code that follows into the C# source file.
3. Edit the program to use the VISA address of your oscilloscope.
4. Add References.

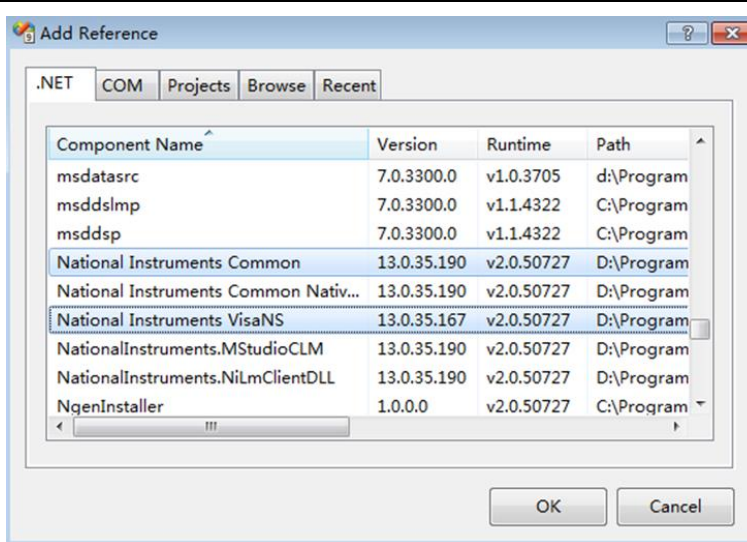
Add Ivi.Visa.dll and NationalInstruments.Visa.dll to the project.

If your NI-VISA version is too low(e.g. 5.4.0),you should add NationalInstruments.Common.dll and NationalInstruments.VisaNS.dll to the project. (Item 11 of this link details some of the main differences between NI-VISA .NET and NI-VISA NS: [NI-VISA .NET Library - NI](#))

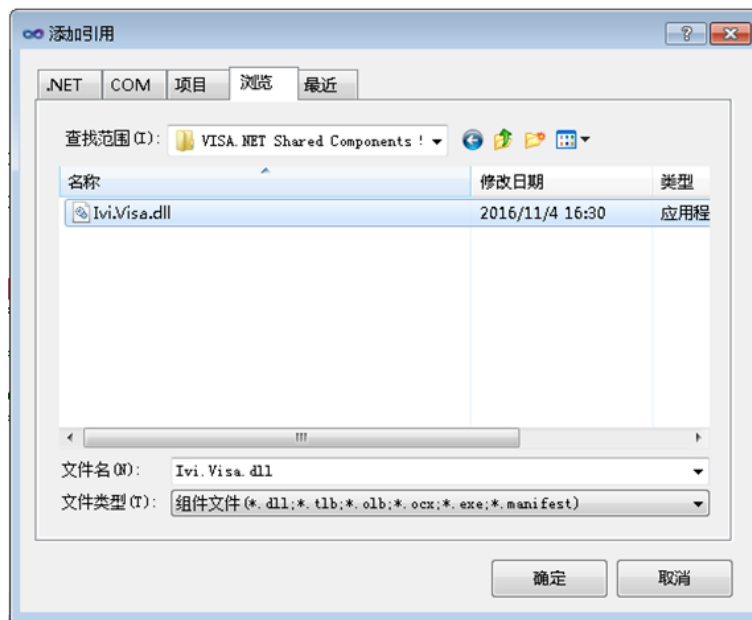
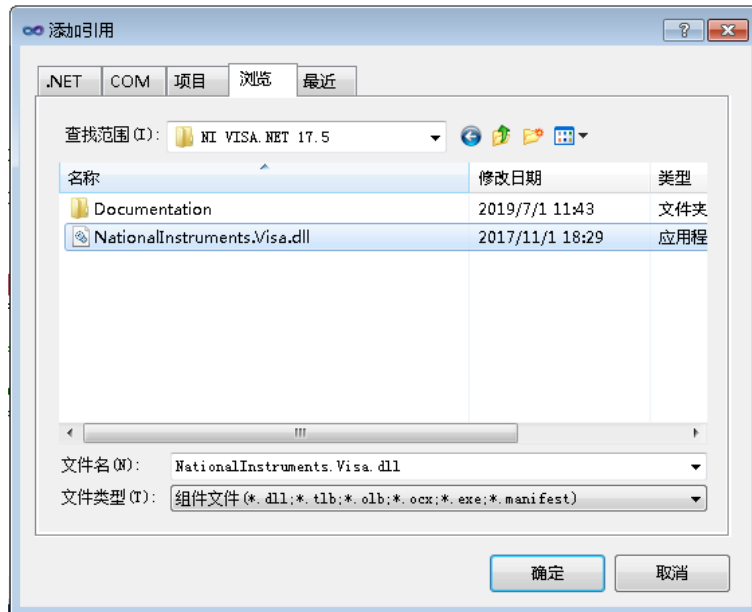
(Notice: you must install the .NET Framework 3.5/4.0/4.5 Languages support when you install the NI-VISA.)

- Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
- Choose Add Reference....
- In the Add Reference dialog, select the Browse tab, and navigate to the NI-VISA installed folder. (for example: C:\Program Files (x86)\IVI Foundation\VISA\Microsoft.NET\..)
- Select the .dll file above; then, click OK.

Add references based on VisaNS in Visual Studio 2008



Add references based on NI-VISA.NET in Visual Studio 2010



```

5. Code on VisaNS:
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using NationalInstruments.VisaNS;

namespace TestVisa
{
class Program
    
```

```
{
static void Main(string[] args)
{
// Find all the USBTMC resources
string[]
usbRsrcStrings=ResourceManager.GetLocalManager().FindResources("USB?*INSTR");
if (usbRsrcStrings.Length <= 0)
{
Console.WriteLine("Cannot find USBTMC Device!");
return;
}

//Choose the first resource string to connect the device.
//You can input the address manually
//USBTMC:
//MessageBasedSession
mbSession=(MessageBasedSession)ResourceManager.GetLocalManager().Open("USB0::0xF
4EC::0xEE38::0123456789::INSTR");
//TCP IP:
//MessageBasedSession
mbSession=(MessageBasedSession)ResourceManager.GetLocalManager().Open("TCPIP0::1
92.168.1.100::INSTR");
MessageBasedSession
mbSession=(MessageBasedSession)ResourceManager.GetLocalManager().Open(usbRsrcStri
ngs[0]);
mbSession.Write("*IDN?");
string result = mbSession.ReadString();
mbSession.Dispose();
Console.WriteLine(result);
}
}
}
```

6. Code on Visa.NET:

```
using System;
using System.Collections.Generic;
using System.Linq;
using NationalInstruments.Visa;
```

```

using Ivi.Visa;

namespace test_visa_csharp
{
    static class Program
    {
        static void Main()
        {
            TcpipSession section = new TcpipSession("TCPIP::10.12.255.135::inst0::INSTR");
            IMessageBasedFormattedIO io = section.FormattedIO;
            io.WriteLine("*IDN?");
            string result = io.ReadLine();
            section.Dispose();
            Console.WriteLine(result);
        }
    }
}

```

#### 7. Example Read Waveform on Visa.NET:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using NationalInstruments.Visa;
using Ivi.Visa;
using System.Runtime.InteropServices;
using System.Threading;

[StructLayout(LayoutKind.Sequential, Pack = 2)]
public struct RT_TIME_OLD
{
    public double seconds; //8
    public char minutes; //1
    public char hours; //1
    public char days; //1
    public char months; //1
}

```

```
public short year;    //2
public short dummy;  //2
};

[StructLayout(LayoutKind.Sequential, Pack = 2)]
public struct WD_PARAM
{
    /* MANDATORY PART */
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
    public string descriptor_name;    /* will contain "WAVEDESC" char [16] */
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
    public string template_name; //char [16]

    public short comm_type;
    public short comm_order;

    /* DESCRIPTOR PART */
    public Int32 wave_desc_length; //4
    public Int32 user_text_length;
    public Int32 res_desc1;

    public Int32 trig_time_array;
    public Int32 ris_time_array;
    public Int32 res_array1;

    /* ARRAY PART */
    public Int32 wave_array_1;
    public Int32 wave_array_2;    /* this is 0 if not present */
    public Int32 res_array2;
    public Int32 res_array3;

    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
    public string instrument_name;//char [16]

    public UInt32 instrument_number; //4

    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
```

```
public string trace_label; //char [16]

public Int32 reserved_data_count; /* Equal to internalDescriptor.data_count. */
/* Currently not documented to users. */
/* Necessary because */
/* internalDescriptor.dat_count can't be */
/* recomputed from other fields of */
/* externalDescriptor; */
/* wrong internalDescriptor.data_count */
/* is detrimental to automated testing */
/* of wf? / wf and card:sto / rec. */

/* The following variables describe the waveform type and the time at
   which the waveform was generated.*/

public Int32 wave_array_count; /* actual nbr of data items in simple array
                               = nominal + extra points required. This may
                               be necessary for expansion but should NOT
                               be used for calculations --> use last_valid
                               - first_valid for determining the number
                               of points in computations and display */

public Int32 points_per_screen; /* nominal number of points in the waveform */

public Int32 first_valid; /* ; count of number of points to skip
                           ;; before first good point
                           ;; FIRST_VALID_POINT = 0
                           ;; for normal waveforms. */

/* index to first valid point in data array.
   Point 0 of the data array always maps to
   the first pixel before the left edge of the
   screen. This means that if the waveform
   does not actual start at that edge, the data
```



in the beginning of the array is invalid -  
however, no assumption is made that it is  
zero. For an unexpanded waveform, this  
is set to 0 \*/

```
public Int32 last_valid;    /* ; index of last good data point
                           ;; in record before padding (blanking)
                           ;; was started.
                           ;; LAST_VALID_POINT = WAVE_ARRAY_COUNT-1
                           ;; except for aborted sequence
                           ;; and rollmode acquisitions    */

public Int32 first_point;

public Int32 sparsing_factor;

public Int32 segment_no;

public Int32 subarray_count; /* for Sequence, acquired segment count,
                              between 0 and NOM_SUBARRAY_COUNT */

public Int32 sweeps_per_acq; /* for Averages and Extrema:
                              number of sweeps accumulated */

public short points_per_pair; /* for Peak Detect only */
public short pair_offset;    /* for Peak Detect only */

public float vertical_gain; /* total gain of waveform, units per lsb */

public float vertical_offset; /* total vertical offset of waveform */

public float code_per_div;   // float representation of the max integer (byte or word) value
used
// corresponds to verFrameStop
public float reserved;      // float representation of the min integer (byte or word)
value used
// corresponds to verFrameStart
```

```
public short nominal_bits; /* estimated: Ch1, Ch2: 8; averaging: 12, etc */

public short nom_subarray_count; /* for Sequence, nominal segment count
                                   else 1 */

public float horizontal_interval; /* this corresponds to the sampling interval
                                   (ie. time/point) for time domain waveforms
                                   and freq/point for FFT's. It is the nominal
                                   time between successive points in the data
                                   In RIS, it is the equivalent sampling rate*/

public double horizontal_offset; /* this corresponds to trigger offset in time
                                   domain for zero'th sweep of trigger,
                                   seconds from trigger to zero'th data point
                                   (ie. actual trigger delay) */

public double pixel_offset; /* from trigger to zero'th pixel of display
                              segment in time domain. measured in seconds
                              (ie. nominal trigger delay) */

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 48)]
public string vertunit; //char [48]
[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 48)]
public string horunit; //char [48]

public float horiz_uncertainty; // this correspond to the TDC resolution
// in seconds.
public RT_TIME_OLD trigger_time; /* also for sequence waveforms */

public float acq_duration; /* time in seconds; for sequence & RIS */

public short ca_record_type; /* type of waveform (see enum above) */

public short processing_done; /* indication of whether any processing done
                               see enum (or bit pattern) above */
```

```
public short reserved5;

public short ris_sweeps;    /* ; for RIS, the number of sweeps
                           ;; else 1                */

/* the following information should be for history only */

public short time_base;    /* this is the enumerated time/div */
public short vertical_coupling;

public float probe_attenuation;

public short fixed_vertical_gain;
public short band_width_limit;

public float vertical_vernier;    /* needed for waveform display */
public float acquisition_vertical_offset; /* divisions */

public short wave_source;
};

namespace test_visa_csharp
{
    static class Program
    {
        /// <summary>
        /// </summary>
        [STAThread]
        static void Main()
        {
            /*
            //Connect Via TCPIP
            TcpipSession section = new TcpipSession("TCPIP::10.12.58.1::inst0::INSTR");
            IMessageBasedFormattedIO io = section.FormattedIO;
            io.WriteLine("*IDN?");
            string result = io.ReadLine();
            section.Dispose();
            */
        }
    }
}
```

```
    Console.WriteLine(result);
    */
    //Connect Via TCPIP or USB
    var rmSession = new ResourceManager();
    MessageBasedSession mbSession =
(MessageBasedSession)rmSession.Open("TCPIP::10.12.60.1::inst0::INSTR");
    sds_initialize(mbSession);
    mbSession.RawIO.Write("TRIG:MODE SINGLE");
    bool state = WaitAcquisitionComplete(mbSession);
    if (state)
    {
        float[] c1_volt = sds_fetchwaveform(mbSession,"C1");
    }
    mbSession.Dispose();
}

public static void sds_initialize(MessageBasedSession mbSession)
{
    mbSession.RawIO.Write("*IDN?");
    string result = mbSession.RawIO.ReadString();
    mbSession.RawIO.Write("CHDR OFF");
    Console.WriteLine(result);
}

public static bool WaitAcquisitionComplete(MessageBasedSession mbSession)
{
    mbSession.RawIO.Write("TRIG:MODE?");
    string mode = mbSession.RawIO.ReadString();
    if (mode.Contains("SINGle"))
    {
        while (true)
        {
            mbSession.RawIO.Write("TRIG:STAT?");
            string result = mbSession.RawIO.ReadString();
            if (result.Contains("Stop"))
            {
                Console.WriteLine("Single Acquisition finished");
                return true;
            }
        }
    }
}
```

```
        }
    }
}
else
{
    while (true)
    {
        mbSession.RawIO.Write("INR?");
        string result = mbSession.RawIO.ReadString();
        Int16 state = Convert.ToInt16(result);
        if ((state & 0x01) == 1)
        {
            Console.WriteLine("Acquisition finished");
            return true;
        }
    }
}
return false;
}

public static float[] sds_fetchwaveform(MessageBasedSession mbSession,string channel)
{
    string src_cmd = string.Format("WAV:SOUR {0}", channel);
    mbSession.RawIO.Write(src_cmd);
    mbSession.RawIO.Write("WAV:STAR 0");
    mbSession.RawIO.Write("WAV:PRE?");
    byte[] WaveParamBytes = mbSession.FormattedIO.ReadBinaryBlockOfByte();
    var wp = ConvertToWaveFormParam(WaveParamBytes);
    Console.WriteLine(wp.code_per_div);
    mbSession.Clear();

    mbSession.RawIO.Write("WAVEform:MAXPoint?");
    string result = mbSession.RawIO.ReadString();
    float one_piece_num = (float)Convert.ToSingle(result);
    mbSession.RawIO.Write("ACQ:POIN?");
    result = mbSession.RawIO.ReadString();
    int point = (int)Convert.ToSingle(result);
}
```

```

Int16[] WaveDataInt16All = new Int16[point];
sbyte[] WaveDataAll = new sbyte[point];
float[] volt = new float[WaveDataAll.Length];

if (wp.nominal_bits > 8)
{
    WaveDataInt16All = GetWaveData16bit(mbSession, point, one_piece_num);
    for (int i = 0; i < WaveDataInt16All.Length; i++)
    {
        volt[i] = (WaveDataInt16All[i] / wp.code_per_div * wp.vertical_gain -
wp.vertical_offset) * wp.probe_attenuation;
    }
}
else
{
    WaveDataAll = GetWaveData8bit(mbSession, point, one_piece_num);
    for (int i = 0; i < WaveDataAll.Length; i++)
    {
        volt[i] = (WaveDataAll[i] / wp.code_per_div * wp.vertical_gain - wp.vertical_offset) *
wp.probe_attenuation;
    }
}
return volt;
}

```

```

public static sbyte[] GetWaveData8bit(MessageBasedSession mbSession, int point, float
one_piece_num)
{
    sbyte[] WaveDataAll = new sbyte[point];

    int read_times = (int)System.Math.Ceiling((point / one_piece_num));
    mbSession.RawIO.Write("WAV:WIDT BYTE");
    DateTime dt1 = DateTime.Now;
    for (int i = 0; i < read_times; i++)
    {
        int start = (int)(i * one_piece_num);
        string start_cmd = string.Format("WAVEform:STARt {0}", start);
        mbSession.RawIO.Write(start_cmd);
    }
}

```

```
        DateTime dt3 = DateTime.Now;
        mbSession.RawIO.Write("WAV:DATA?");
        sbyte[] WaveDataBytes = mbSession.FormattedIO.ReadBinaryBlockOfSByte();
        Console.WriteLine(WaveDataBytes.Length);
        Array.Copy(WaveDataBytes, 0, WaveDataAll, start, WaveDataBytes.Length);
    }
    mbSession.Clear();
    DateTime dt2 = DateTime.Now;
    TimeSpan ts = dt2.Subtract(dt1);
    Console.WriteLine("used time = {0}ms", ts.TotalMilliseconds);
    Console.WriteLine("WaveDataAll.Length = {0}", WaveDataAll.Length);
    return WaveDataAll;
}
```

```
public static Int16[] GetWaveData16bit(MessageBasedSession mbSession, int point, float
one_piece_num)
{
    Int16[] WaveDataInt16All = new Int16[point];

    mbSession.RawIO.Write("WAV:WIDT WORD");
    mbSession.RawIO.Write("WAV:BYTEORDER MSB");
    int read_times = (int)System.Math.Ceiling((point / one_piece_num));
    DateTime dt1 = DateTime.Now;
    for (int i = 0; i < read_times; i++)
    {
        int start = (int)(i * one_piece_num);
        string start_cmd = string.Format("WAVEform:STARt {0}", start);
        mbSession.RawIO.Write(start_cmd);
        mbSession.RawIO.Write("WAV:DATA?");
        Int16[] WaveDataBytes = mbSession.FormattedIO.ReadBinaryBlockOfInt16();
        Array.Copy(WaveDataBytes, 0, WaveDataInt16All, start, WaveDataBytes.Length);
    }
    mbSession.Clear();
    DateTime dt2 = DateTime.Now;
    TimeSpan ts = dt2.Subtract(dt1);
    Console.WriteLine("used time = {0}ms", ts.TotalMilliseconds);
    Console.WriteLine("WaveDataInt16All.Length = {0}", WaveDataInt16All.Length);
    return WaveDataInt16All;
}
```

```
    }  
  
    public static WD_PARAM ConvertToWaveFormParam(byte[] parambuff)  
    {  
        int strucsize = Marshal.SizeOf(typeof(WD_PARAM));  
        IntPtr ptemp = Marshal.AllocHGlobal(strucsize);  
        Marshal.Copy(parambuff, 0, ptemp, strucsize);  
        WD_PARAM wd = (WD_PARAM)Marshal.PtrToStructure(ptemp, typeof(WD_PARAM));  
        Marshal.FreeHGlobal(ptemp);  
        return wd;  
    }  
}
```



## Examples of Using Sockets

Socket communication is a basic communication technology in computer network. It allows applications to communicate through the standard network protocol mechanism built by network hardware and operation system.

This method is a two-way communication between the instrument and the computer through a fixed port number.

Note that SCPI strings are terminated with a “\n” (new line) character.

### Python Example

Python has a low-level networking module that provides access to the socket interface. Python scripts can be written for sockets to do a variety of test and measurement tasks.

**Environment:** Windows7 32-bit, Python v2.7.5

**Description:** Open a socket, send a query, and repeat this loop for 10 times, finally close the socket.

Below is the code of the script:

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
#-----
# The short script is a example that open a socket, sends a query,
# print the return message and closes the socket.
#-----
import socket # for sockets
import sys # for exit
import time # for sleep
#-----
remote_ip = "10.12.255.209" # should match the instrument's IP address
port = 5025 # the port number of the instrument service
count = 0

def SocketConnect():
    try:
```

```
#create an AF_INET, STREAM socket (TCP)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error:
    print ('Failed to create socket.')
    sys.exit();
try:
    #Connect to remote server
    s.connect((remote_ip , port))
except socket.error:
    print ('failed to connect to ip ' + remote_ip)
return s

def SocketQuery(Sock, cmd):
    try :
        #Send cmd string
        Sock.sendall(cmd)
        Sock.sendall(b'\n')
        time.sleep(1)
    except socket.error:
        #Send failed
        print ('Send failed')
        sys.exit()
    reply = Sock.recv(4096)
    return reply

def SocketClose(Sock):
    #close the socket
    Sock.close()
    time.sleep(.300)

def main():
    global remote_ip
    global port
    global count

    # Body: send the SCPI commands *IDN? 10 times and print the return message
    s = SocketConnect()
    for i in range(10):
```

```
qStr = SocketQuery(s, b'*IDN?')
print (str(count) + ":: " + str(qStr))
count = count + 1
SocketClose(s)
input('Press "Enter" to exit')

if __name__ == '__main__':
    proc = main()
```

## C Example

```
int MySocket;
if((MySocket=socket(PF_INET,SOCK_STREAM,0))==-1)
{
    exit(1);
}
struct in_addr
{
    unsigned long s_addr;
};
struct sockaddr_in
{
    short int sin_family; // Address family
    unsigned short int sin_port; // Port number
    struct in_addr sin_addr; // Internet address
    unsigned char sin_zero[8]; // Padding
};
struct sockaddr_in MyAddress;

// Initialize the whole structure to zero
memset(&MyAddress,0,sizeof(struct sockaddr_in));
// Then set the individual fields
MyAddress.sin_family=PF_INET; // IPv4
MyAddress.sin_port=htons(5025); // Port number used by most instruments
MyAddress.sin_addr.s_addr=inet_addr(ntsaddr_in); // IP Address

// Establish TCP connection
if(connect(MySocket,(struct sockaddr*)&MyAddress,sizeof(struct sockaddr_in))==-1)
{
    exit(1);
}

// Send SCPI command
if(send(MySocket,ands,sizeof(t_addr(
{
    exit(1);
}
```


```
// Read response
char buffer[200];
int actual;
if((actual=recv(MySocket,&buffer[0],200,0))==-1)
{
    exit(1);
}
buffer[actual]= 0; // Add zero character (C string)
printf(d zero character (C string)],2
```

```
// Close socket
if(close(MySocket)==-1)
{
    exit(1);
}
```

## Common Command Examples

This section lists the programming instances of common commands.

**Environment:** Windows7 32-bit, Python v3.6.5, pyvisa-1.9, Matplotlib-3.1.1

|   |  |
|---|--|
|  | <p><b>Note:</b> When using the visa library, you should pay attention to the following settings:</p> <ul style="list-style-type: none"> <li>➤ Set the I/O buffer size.<br/>I.E. For the command “:WAVEform:DATA?”, the read buffer size depends on the number of waveform points. When it needs to read in segments, the size of each segment is vary from the models.</li> <li>➤ Set the timeout value.<br/>The timeout value is related to the network speed or USB transmission speed. Please evaluate by yourself. The initial value is generally 2s.</li> </ul> |
|---|--|

### Read Waveform Data Example

```

# Import modules.

# -----
import visa
import pylab as pl
import struct
import math
import gc

# Global variables
# (Modify the following global variables according to the model).
# -----
SDS_RSC = "TCPIP0::10.12.59.1::inst0::INSTR"
CHANNEL = "C2"
HORI_NUM = 10
tdiv_enum = [200e-12,500e-12, 1e-9,\
             2e-9, 5e-9, 10e-9, 20e-9, 50e-9, 100e-9, 200e-9, 500e-9, \
             1e-6, 2e-6, 5e-6, 10e-6, 20e-6, 50e-6, 100e-6, 200e-6, 500e-6, \
             1e-3, 2e-3, 5e-3, 10e-3, 20e-3, 50e-3, 100e-3, 200e-3, 500e-3, \
             1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]

# =====
# main_desc:Analyzing waveform parameters from data blocks
# =====

def main_desc(recv):
    param_addr_type={"data_bytes":[0x3c,"i"],

```

```
    "point_num":[0x74,'i'],
    "fp":[0x84,'i'],
    "sp":[0x88,'i'],
    "vdiv":[0x9c,'f'],
    "offset":[0xa0,'f'],
    "code":[0xa4,'f'],
    "adc_bit":[0xac,'h'],
    "interval":[0xb0,'f'],
    "delay":[0xb4,'d'],
    "tdiv":[0x144,'h'],
    "probe":[0x148,'f']}
data_byte = {"i": 4, "f": 4, "h": 2, "d": 8}
param_val ={}

for key,addr_type in param_addr_type.items():
    addr_start = addr_type[0]
    format = addr_type[1]
    bytes = recv[addr_start:addr_start+data_byte[format]]
    param_val[key] = struct.unpack(format, bytes)[0]
param_val["tdiv"] = tdiv_enum[param_val["tdiv"]]
param_val["vdiv"] = param_val["vdiv"]*param_val["probe"]
param_val["offset"] = param_val["offset"]*param_val["probe"]
return param_val

# =====
# Main program:
# =====
def main_wf_data():
    _rm = visa.ResourceManager()
    sds = _rm.open_resource(SDS_RSC)
    sds.timeout = 2000 # default value is 2000(2s)
    sds.chunk_size = 20 * 1024 * 1024 # default value is 20*1024(20k bytes)

    # Get the channel waveform parameter data blocks and parse them
    sds.write(":WAVeform:START 0")
    sds.write("WAV:SOUR {}".format(CHANNEL))
    sds.write("WAV:PREAmble?")
    recv_all = sds.read_raw()
```

```

recv = recv_all[recv_all.find(b'#') + 11:]
print(len(recv))
vdiv, ofst, interval, trdl, tdiv, vcode_per, adc_bit = main_desc(recv)
print(vdiv, ofst, interval, trdl, tdiv, vcode_per, adc_bit)

# Get the waveform points and confirm the number of waveform slice reads
points = param_dic["point_num"]
one_piece_num = float(sds.query(":WAVeform:MAXPoint?").strip())
read_times = math.ceil(points / one_piece_num)
#Set the number of read points per slice, if the waveform points is greater than the maximum
number of slice reads
if points > one_piece_num:
    sds.write(":WAVeform:POINT {}".format(one_piece_num))
# Choose the format of the data returned
sds.write(":WAVeform:WIDTH BYTE")
if param_dic["adc_bit"] > 8:
    sds.write(":WAVeform:WIDTH WORD")

#Get the waveform data for each slice
recv_byte = b"
for i in range(0, read_times):
    start = i * one_piece_num
    #Set the starting point of each slice
    sds.write(":WAVeform:START {}".format(start))
    #Get the waveform data of each slice
    sds.write("WAV:DATA?")
    recv_rtn = sds.read_raw()
    #Splice each waveform data based on data block information
    block_start = recv_rtn.find(b'#')
    data_digit = int(recv_rtn[block_start + 1:block_start + 2])
    data_start = block_start + 2 + data_digit
    data_len = int(recv_rtn[block_start + 2:data_start])
    recv_byte += recv_rtn[data_start:data_start + data_len]
# Unpack signed byte data.
if param_dic["adc_bit"] > 8:
    convert_data = struct.unpack("%dh"%points, recv_byte)
else:
    convert_data = struct.unpack("%db"%points, recv_byte)

```



```
del recv_byte
gc.collect()
#Calculate the voltage value and time value
time_value = []
volt_value = []
for idx in range(0, len(convert_data)):
    volt_value.append(convert_data[idx]/param_dic["code"]*param_dic["vdiv"]-
param_dic["offset"])
    time_data = - (param_dic["tdiv"] * HORI_NUM / 2) + idx * param_dic["interval"] +
param_dic["delay"]
    time_value.append(time_data)
print(len(volt_value))
#Draw Waveform
pl.figure(figsize=(7, 5))
pl.plot(time_value, volt_value, markersize=2, label=u"Y-T")
pl.legend()
pl.grid()
pl.show()

if __name__ == '__main__':
    main_wf_data()
```

## Read Waveform Data of Digital Example

```

    # Import modules.
# -----
import visa
import pylab as pl
import struct

# Global variables
# (Modify the following global variables according to the model).
# -----
SDS_RSC = "TCPIP0::10.12.59.1::inst0::INSTR"
CHANNEL = "D0"
HORI_NUM = 10
tdiv_enum = [100e-12, 200e-12, 500e-12, \
             1e-9, 2e-9, 5e-9, 10e-9, 20e-9, 50e-9, 100e-9, 200e-9, 500e-9, \
             1e-6, 2e-6, 5e-6, 10e-6, 20e-6, 50e-6, 100e-6, 200e-6, 500e-6, \
             1e-3, 2e-3, 5e-3, 10e-3, 20e-3, 50e-3, 100e-3, 200e-3, 500e-3, \
             1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]

# =====
# get_char_bit: Get each bit from a byte
# =====
def get_char_bit(char,n):
    return (char >> n) & 1

# =====
# main_desc: Analyzing waveform parameters from data blocks
# =====
def main_desc(recv):
    param_addr_type = {"data_bytes": [0x3c, 'i'],
                      "point_num": [0x74, 'i'],
                      "interval": [0xb0, 'f'],
                      "delay": [0xb4, 'd'],
                      "tdiv": [0x144, 'h']}
    data_byte = {"i": 4, "f": 4, "h": 2, "d": 8}
    param_val = {}

    for key, addr_type in param_addr_type.items():

```

```
    addr_start = addr_type[0]
    format = addr_type[1]
    bytes = recv[addr_start:addr_start + data_byte[format]]
    param_val[key] = struct.unpack(format, bytes)[0]
    param_val["tdiv"] = tdiv_enum[param_val["tdiv"]]
return param_val
```

```
# =====
# Main program:
# =====
```

```
def main_digital_wf_data():
    _rm = visa.ResourceManager()
    sds = _rm.open_resource(SDS_RSC)
    # Get the digital channel waveform parameter data blocks and parse them
    sds.write("WAV:SOUR {}".format(CHANNEL))
    sds.write("WAV:PREAmble?")
    recv_all = sds.read_raw()
    recv = recv_all[recv_all.find(b'#')+11:]
    param_val = main_desc(recv)
    print(param_val)
    # Get the waveform data
    sds.write("WAV:DATA?")
    recv_rtn = sds.read_raw()
    block_start = recv_rtn.find(b'#')
    data_digit = int(recv_rtn[block_start + 1:block_start + 2])
    data_start = block_start + 2 + data_digit
    data_len = int(recv_rtn[block_start + 2:data_start])
    data = recv_rtn[data_start:data_start + data_len]
    # Calculate the voltage value and time value
    volt_value = []
    for char in data:
        for i in range(0,8):
            volt_value.append(get_char_bit(char,i))
    print(len(volt_value))
    time_value = []
    for idx in range(0,len(volt_value)):
        time_data = - (param_val["tdiv"] * HORI_NUM / 2) + idx * param_val["interval"] +
param_val["delay"]
```

```
    time_value.append(time_data)
# Draw Waveform
pl.figure(figsize=(7,5))
pl.ylim(-1,2)
pl.plot(time_value,volt_value,markersize=2,label=u"Y-T")
pl.legend()
pl.grid()
pl.show()

if __name__=='__main__':
    main_digital_wf_data()
```

## Read Waveform Data of FFT Example

```
# Import modules.
# -----

import visa
import pylab as pl
import struct
import math
import gc

# Global variables
# -----

SDS_RSC = "TCPIP0::10.12.255.127::inst0::INSTR"
FUNC = "FUNC1"

# =====
# main_desc:Analyzing waveform parameters from data blocks
# =====

def main_desc(recv):
    param_addr_type={"data_bytes":[0x3c,'i'],
                    "point_num":[0x74,'i'],
                    "fp":[0x84,'i'],
                    "sp":[0x88,'i'],
                    "vdiv":[0x9c,'f'],
                    "offset":[0xa0,'f'],
                    "code":[0xa4,'f'],
                    "adc_bit":[0xac,'h'],
                    "interval":[0xb0,'f'],
                    "delay":[0xb4,'d'],
                    "tdiv":[0x144,'h'],
                    "probe":[0x148,'f']}
    data_byte = {"i": 4, "f": 4, "h": 2, "d": 8}
    param_val ={}

    for key,addr_type in param_addr_type.items():
        addr_start = addr_type[0]
        format = addr_type[1]
        bytes = recv[addr_start:addr_start+data_byte[format]]
        param_val[key] = struct.unpack(format, bytes)[0]
```

```

param_val["tdiv"] = tdiv_enum[param_val["tdiv"]]
param_val["vdiv"] = param_val["vdiv"]*param_val["probe"]
param_val["offset"] = param_val["offset"]*param_val["probe"]
return param_val

# =====
# Main program:
# =====

def main_fft_data():
    _rm = visa.ResourceManager()
    sds = _rm.open_resource(SDS_RSC)
    # Get the channel waveform parameter data blocks and parse them
    sds.write("WAV:SOUR F1")
    sds.write("WAV:PREamble?")
    recv_all = sds.read_raw()
    recv = recv_all[recv_all.find(b'#') + 11:]
    param_val = main_desc(recv)
    display_len = int(param_val["delay"]/ param_val["interval"])+1
    unit = sds.query("{}:FFT:UNIT?".format(FUNC)).strip() # {Vrms,DBm,DBVrms}
    if unit == "DBm":
        load = float(sds.query("{}:FFT:LOAD?".format(FUNC)).strip())
    mode = sds.query("{}:FFT:MODE?".format(FUNC)).strip() #
    {NORMal|MAXHold|AVERage[,num]}
    # Get the waveform data
    sds.write("WAV:DATA?")
    recv_all = sds.read_raw().rstrip()
    block_start = recv_all.find(b'#')
    data_digit = int(recv_all[block_start + 1:block_start + 2])
    data_start = block_start + 2 + data_digit
    recv = recv_all[data_start:]
    print(len(recv))
    # Unpack data.
    volt_value = []
    freq_value = []
    len_data = int(len(recv) / 8) ##采样定理 f/2
    print(len_data)
    print(recv[0:4])
    for i in range(0, len_data):

```

```
data_rel = struct.unpack("f", recv[8 * i:8 * i + 4])
data_imag = struct.unpack("f", recv[8 * i + 4:8 * i + 8])
data_rel = list(data_rel)[0]
data_imag = list(data_imag)[0]
if mode == "NORMal":
    data_float = math.sqrt(pow(float(data_rel), 2) + pow(float(data_imag), 2))
else:
    data_float = float(data_rel)
if unit == "DBVrms":
    data_float = 20*math.log10(data_float)
elif unit == "DBm":
    data_float = 10 * math.log10(data_float*data_float/load/1E-3)
    volt_value.append(data_float)
    freq_value.append(i*param_val["interval"])
# Draw Waveform
pl.figure(figsize=(7, 5))
pl.plot(freq_value, volt_value, markersize=2)
pl.legend()
pl.grid()
pl.show()

if __name__ == '__main__':
    main_fft_data()
```

**Read Sequence Waveform Data Example**

```

# Import modules.
# -----
import visa
import pylab as pl
import time as t
import math
import struct
import gc

# Global variables
# (Modify the following global variables according to the model).
# -----
CHANNEL = "C2"
HORI_NUM = 10
TDIV_ENUM = [100e-12, 200e-12, 500e-12, \
             1e-9, 2e-9, 5e-9, 10e-9, 20e-9, 50e-9, 100e-9, 200e-9, 500e-9, \
             1e-6, 2e-6, 5e-6, 10e-6, 20e-6, 50e-6, 100e-6, 200e-6, 500e-6, \
             1e-3, 2e-3, 5e-3, 10e-3, 20e-3, 50e-3, 100e-3, 200e-3, 500e-3, \
             1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]

# =====
# main_wf_desc:Analyzing waveform parameters from data blocks
# =====
def main_wf_desc(recv):
    param_addr_type = {"width":[0x20,"h"],#01-16bit,00-8bit
                       "order":[0x22,"h"],#01-MSB,00-LSB
                       "data_bytes": [0x3c, "i"],
                       "one_frame_pts":[0x74, 'i'],#pts of single frame,maybe bigger than 12.5M
                       "fp": [0x84, 'i'],
                       "sp": [0x88, 'i'],
                       "sn":[0xae,"h"],
                       "read_frame":[0x90,"i"],#all sequence frames number return by this command
                       "sum_frame":[0x94,"i"],#all sequence frames number acquired
                       "vdiv": [0x9c, 'f'],
                       "offset": [0xa0, 'f'],
                       "code": [0xa4, 'f'],
                       "adc_bit": [0xac, 'h'],

```



```
        "interval": [0xb0, 'f'],
        "delay": [0xb4, 'd'],
        "tdiv": [0x144, 'h'],
        "probe": [0x148, 'f']}
data_byte = {"i": 4, "f": 4, "h": 2, "d": 8}
param_val = {}
for key, addr_type in param_addr_type.items():
    addr_start = addr_type[0]
    format = addr_type[1]
    bytes = recv[addr_start:addr_start + data_byte[format]]
    param_val[key] = struct.unpack(format, bytes)[0]
param_val["tdiv"] = TDIV_ENUM[param_val["tdiv"]]
param_val["vdiv"] = param_val["vdiv"] * param_val["probe"]
param_val["offset"] = param_val["offset"] * param_val["probe"]
    return param_val

# =====
# main_time_stamp_deal: Parsing timestamps from binary blocks
# =====

def main_time_stamp_deal(time):
    seconds = time[0x00:0x08]    # type:long double
    minutes = time[0x08:0x09]   # type:char
    hours = time[0x09:0x0a]     # type:char
    days = time[0x0a:0x0b]      # type:char
    months = time[0x0b:0x0c]    # type:char
    year = time[0x0c:0x0e]      # type:short

    seconds = struct.unpack('d',seconds)[0]
    minutes = struct.unpack('c', minutes)[0]
    hours = struct.unpack('c', hours)[0]
    days = struct.unpack('c', days)[0]
    months = struct.unpack('c', months)[0]
    year = struct.unpack('h', year)[0]
    months = int.from_bytes(months, byteorder='big', signed=False)
    days = int.from_bytes(days, byteorder='big', signed=False)
    hours = int.from_bytes(hours, byteorder='big', signed=False)
    minutes = int.from_bytes(minutes, byteorder='big', signed=False)
    print("{}{/}/{}:0:0:0".format(year,months,days,hours,minutes,seconds))
```

```

# =====
# Main program: Read data of all sequence frame.
# When total points num (single_frame_pts * frame_num) is
# bigger than 12.5Mpts, you have to read more than one time.
# Frames number and points number readed this time will be
# saved in the head parameter, see main_wf_desc.
# =====

def main_all_frame(sds):
    sds.write(":WAVeform:SOURce {}".format(CHANNEL))
    sds.write(":WAVeform:STARt 0")
    sds.write(":WAVeform:POINt 0")
    sds.write(":WAVeform:SEQUence 0,0")
    sds.timeout = 2000 #default value is 2000(2s)
    sds.chunk_size = 20*1024*1024 #default value is 20*1024(20k bytes)

    sds.write(":WAVeform:PREAmble?")
    recv_all = sds.read_raw()
    recv = recv_all[recv_all.find(b'#')+11:]
    print(len(recv))
    param_dic = main_wf_desc(recv)
    read_times = math.ceil(param_dic["sum_frame"]/param_dic["read_frame"])
    print("read_times=",read_times)
    one_piece_num = float(sds.query(":WAVeform:MAXPoint?").strip())

    for i in range(0,read_times):
        if i+1 == read_times: :#frame num of last read time
            read_frame = param_dic["sum_frame"] -(read_times-1)*param_dic["read_frame"]
        else:
            read_frame = param_dic["read_frame"]
        sds.write(":WAVeform:SEQUence {},{}".format(0,read_frame*i+1))
        sds.write(":WAVeform:PREAmble?")
        recv_rtn = sds.read_raw()
        recv_desc = recv_rtn[recv_rtn.find(b'#')+11:]
        time_stamp = recv_desc[346:]

        if param_dic["adc_bit"] > 8:
            sds.write(":WAVeform:WIDTH WORD")

```

```

sds.write(":WAVeform:DATA?")
recv_rtn = sds.read_raw()
block_start = recv_rtn.find(b'#')
data_digit = int(recv_rtn[block_start + 1:block_start + 2])
data_start = block_start + 2 + data_digit
data_len = int(recv_rtn[block_start + 2:data_start])
recv = recv_rtn[data_start:data_start + data_len]

for j in range(0, param_dic["read_frame"]):
    time = time_stamp[16*j:16*(j+1)]#timestamp spends 16 bytes
    main_time_stamp_deal(time)
    if adc_bit > 8:
        start = int(j * param_dic["one_frame_pts"]*2)
        end = int((j + 1) * param_dic["one_frame_pts"]*2)
        convert_data = struct.unpack("%dh" % param_dic["one_frame_pts"], recv[start:end])
    else:
        start = int(j*one_frame_pts)
        end = int((j+1)*one_frame_pts)
        convert_data = struct.unpack("%db" % param_dic["one_frame_pts"], recv[start:end])

    volt_value = []
    time_value = []
    for idx in range(0,len(convert_data)):
        volt_value.append(convert_data[idx] / param_dic["code"] * param_dic["vdiv"] -
param_dic["offset"])
        time_value.append(-(param_dic["tdiv"] * HORI_NUM / 2) + idx * param_dic["interval"] +
param_dic["delay"])

    print('Data convert finish,start to draw!')
    pl.figure(figsize=(7,5))
    pl.plot(time_value,volt_value,markersize=2,label=u"Y-T")
    pl.legend()
    pl.grid()
    pl.show()
    pl.close()
    del volt_value,time_value,convert_data
    gc.collect()

del recv

```

```

gc.collect()

# =====
# Main program: Read data of single frame.
# =====

def main_specify_frame(sds,frame_num):
    sds.write(":WAVeform:SOURce {}".format(CHANNEL))
    sds.write(":WAVeform:STARt 0")
    sds.write(":WAVeform:POINt 0")
    sds.write(":WAVeform:SEQUence {},{}".format(frame_num,0))
    sds.timeout = 2000 # default value is 2000(2s)
    sds.chunk_size = 20 * 1024 * 1024 # default value is 20*1024(20k bytes)

    sds.write(":WAVeform:PREamble?")
    recv_all = sds.read_raw()
    print(len(recv_all))
    recv = recv_all[recv_all.find(b'#')+11:]
    time_stamp = recv[346:]
    main_time_stamp_deal(time_stamp)
    param_dic = main_wf_desc(recv)

    one_piece_num = float(sds.query(":WAVeform:MAXPoint?").strip())
    if param_dic["one_frame_pts"] > one_piece_num:
        sds.write(":WAVeform:POINt {}".format(one_piece_num))
    if param_dic["adc_bit"] > 8:
        sds.write(":WAVeform:WIDTH WORD")

    read_times = math.ceil(param_dic["one_frame_pts"] / one_piece_num)
    data_recv = b""
    for i in range(0, read_times):
        start = i * one_piece_num
        sds.write(":WAVeform:STARt {}".format(start))
        sds.write("WAV:DATA?")
        recv_rtn = sds.read_raw()
        block_start = recv_rtn.find(b'#')
        data_digit = int(recv_rtn[block_start + 1:block_start + 2])
        data_start = block_start + 2 + data_digit
        data_len = int(recv_rtn[block_start + 2:data_start])

```

```
    data_recv += recv_rtn[data_start: data_start + data_len]
print("len(data_recv)=", len(data_recv))
if param_dic["adc_bit"] > 8:
    convert_data = struct.unpack("%dh" % param_dic["one_frame_pts"], data_recv)
else:
    convert_data = struct.unpack("%db" % param_dic["one_frame_pts"], data_recv)

volt_value = []
time_value = []
for idx in range(0, len(convert_data)):
    volt_value.append(convert_data[idx] / param_dic["code"] * param_dic["vdiv"] -
param_dic["offset"])
    time_value.append(-(param_dic["tdiv"] * HORI_NUM / 2) + idx * param_dic["interval"] +
param_dic["delay"])

print("Data convert finish,start to draw!")
pl.figure(figsize=(7, 5))
pl.plot(time_value, volt_value, markersize=2, label=u"Y-T")
pl.legend()
pl.grid()
pl.show()
pl.close()
del volt_value, time_value, data_recv
gc.collect()

if __name__ == '__main__':
    _rm = visa.ResourceManager()
    sds = _rm.open_resource("TCPIP0::10.12.59.1::inst0::INSTR")
    main_all_frame(sds)
    main_specify_frame(sds, 1)
    sds.close()
```

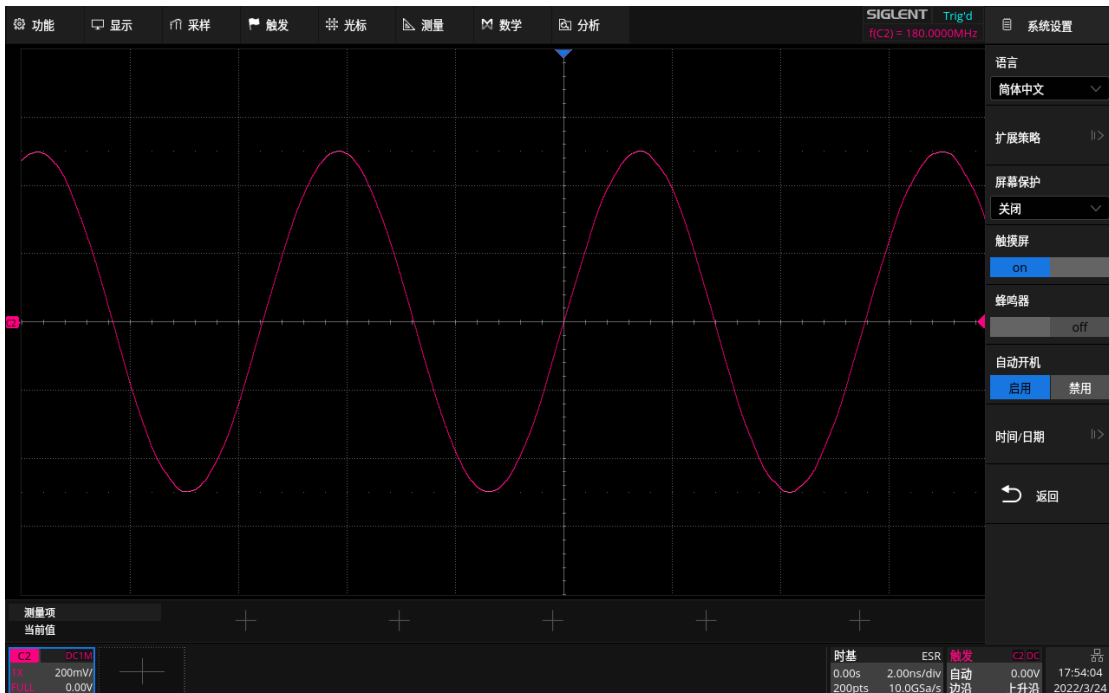
## Screen Dump (PRINT) Example

```
import visa

def main():
    _rm = visa.ResourceManager()
    sds = _rm.open_resource("USB0::0xF4EC::0xEE38::0123456789::INSTR")
    sds.chunk_size = 20*1024*1024 #default value is 20*1024(20k bytes)
    file_name = "F:\\SCDP.bmp"
    sds.write("PRIN? BMP")
    result_str = sds.read_raw()
    f = open(file_name,'wb')
    f.write(result_str)
    f.flush()
    f.close()

if __name__ == '__main__':
    main()
```

Then you can open the file as shown below:





## About SIGLENT

SIGLENT is an international high-tech company, concentrating on R&D, sales, production and services of electronic test & measurement instruments.

SIGLENT first began developing digital oscilloscopes independently in 2002. After more than a decade of continuous development, SIGLENT has extended its product line to include digital oscilloscopes, isolated handheld oscilloscopes, function/arbitrary waveform generators, RF/MW signal generators, spectrum analyzers, vector network analyzers, digital multimeters, DC power supplies, electronic loads and other general purpose test instrumentation. Since its first oscilloscope was launched in 2005, SIGLENT has become the fastest growing manufacturer of digital oscilloscopes. We firmly believe that today SIGLENT is the best value in electronic test & measurement.

### Headquarters:

SIGLENT Technologies Co., Ltd  
Add: Bldg No.4 & No.5, Antongda Industrial  
Zone, 3rd Liuxian Road, Bao'an District,  
Shenzhen, 518101, China  
Tel: + 86 755 3688 7876  
Fax: + 86 755 3359 1582  
Email: [sales@siglent.com](mailto:sales@siglent.com)  
Website: [int.siglent.com](http://int.siglent.com)

### North America:

SIGLENT Technologies America, Inc  
6557 Cochran Rd Solon, Ohio 44139  
Tel: 440-398-5800  
Toll Free: 877-515-5551  
Fax: 440-399-1211  
Email: [info@siglentna.com](mailto:info@siglentna.com)  
Website: [www.siglentna.com](http://www.siglentna.com)

### Europe:

SIGLENT Technologies Germany GmbH  
Add: Staetzlinger Str. 70  
86165 Augsburg, Germany  
Tel: +49(0)-821-666 0 111 0  
Fax: +49(0)-821-666 0 111 22  
Email: [info-eu@siglent.com](mailto:info-eu@siglent.com)  
Website: [www.siglenteu.com](http://www.siglenteu.com)

Follow us on  
Facebook: [SiglentTech](https://www.facebook.com/SiglentTech)

