Tektronix®

TBS1000C Series Digital Oscilloscopes

Programmer Manual

Register now!
Click the following link to protect your product.
tek.com/register

077-1691-02 May 2024

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
US

For product information, sales, service, and technical support visit *tek.com* to find contacts in your area. For warranty information visit *tek.com/warranty*.

Contents

List of Figures....11
List of Tables....12
Getting Started....13
Command Syntax....14
Command Syntax....14
Command and Query Structure....14
Commands....15
Queries....15
Headers in Query Responses....16
Clearing the Output Queue....16
Command Entry....16
Abbreviating Commands....16
Concatenating Commands....16
Message Terminators....17
Constructed Mnemonics....18
Reference Waveform Mnemonics....18
Waveform Mnemonics....18
Cursor Position Mnemonic....18
Measurement Specifier Mnemonics....19
Argument Types....19
Numeric Arguments....19
Quoted String Arguments....20
Block Arguments....20
A commands....22
ACQuire?....22
ACQuire:MAXSamplerate?....22
ACQuire:MODe....22
ACQuire:NUMACq?....23
ACQuire:NUMAVg....24
ACQuire:STATE....24
ACQuire:STOPAfter....25
ALIas....26
ALIas:CATalog?....26
ALIas:DEFine....27
ALIas:DELEte....27
ALIas:DELEte:ALL....28
ALIas:DELEte[:NAMe]....28
ALIas[:STATE]....29
ALLEv?....29
AUTOSet....30
AUTOSet:ENABLE....30
B commands....32
BUSY?....32
C commands....33

*CAL?....33
CALibrate:INTERNAL....33
CALibrate:INTERNal:STARt....34
CALibrate:INTERNal:STATus?....34
CALibrate:RESults?....35
CALibrate:RESults:SPC?....35
CH<x>?....36
CH<x>:AMPSVIAVOLTs:ENABLe....36
CH<x>:AMPSVIAVOLTs :Factor....37
CH<x>:BANdwidth....37
CH<x>:COUPling....38
CH<x>:DESKew....38
CH<x>:INVert....39
CH<x>:LABel....39
CH<x>:OFFSet....40
CH<x>:POSition....40
CH<x>:PRObe....41
CH<x>:PRObe:GAIN....41
CH<x>:PRObe:ID?....42
CH<x>:PRObe:ID:SERnumber?....42
CH<x>:PRObe:ID:TYPE?....43
CH<x>:PRObe:SIGnal....43
CH<x>:PRObe:UNIts?....43
CH<x>:SCAle....44
CH<x>:VOLts....45
CH<x>:YUNit....45
CLEARMenu....46
*CLS....46
CURSor?....47
CURSor:ENABLE....47
CURSor:FUNCtion....48
CURSor:HBArs?....48
CURSor:HBArs:DELTa?....49
CURSor:HBArs:POSITION<x>....49
CURSor:HBArs:UNIts....50
CURSor:HBArs:USE....50
CURSor:MODe....51
CURSor:VBArs?....51
CURSor:VBArs:ALTERNATE<x>?....52
CURSor:VBArs:DELTa?....52
CURSor:VBArs:HPOS<x>?....53
CURSor:VBArs:POSITION<x>....53
CURSor:VBArs:UNIts....54
CURSor:VBArs:VDELTa?....54
CURVe....55
D commands....57
DATa....57
DATa:DESTination....57
DATa:SOUrce....58

DATa:STARt....58
DATa:STOP....59
DATa:WIDth....60
DATE....60
DESE....61
DIAg:FAN....62
DIAg:LOOP:OPTion....62
DIAg:LOOP:OPTion:NTIMes....62
DIAg:LOOP:STOP....63
DIAg:RESUlt:FLAg?....63
DIAg:RESUlt:LOG?....64
DIAg:SELect....64
DIAg:SELect:<function>....65
DIAg:STATE....65
DIAg:TEMPVAL....66
DISplay:GRAticule....66
DISplay:INTENSITy:BACKLight....66
DISplay:PERSistence:STATe....67
DISplay:PERSistence:VALUe....67
E commands....69
ERRLOG:FIRST?....69
ERRLOG:NEXT?....69
*ESE....69
*ESR?....70
EVENT?....70
EVMsg?....71
EVQty?....72
F commands....73
FACtory....73
FEAEN:PASSWORD....73
FFT?....74
FFT:HORizontal:POSition....74
FFT:HORizontal:SCAle....75
FFT:SOURce....75
FFT:SRCWFM....76
FFT:VERTical:POSition....76
FFT:VERTical:SCAle....77
FFT:VERTical:UNIts....77
FFT:VType....77
FFT:WINdow....78
FILESystem?....78
FILESystem:CWD....79
FILESystem:DELEte....79
FILESystem:DIR?....80
FILESystem:FORMat....80
FILESystem:FREESpace?....81
FILESystem:MKDir....81
FILESystem:READFile....82
FILESystem:REName....82

FILESystem:RMDir..... 83
FILESystem:WRITEFile..... 84
FPAnel:PRESS..... 84
FPAnel:TURN..... 85
FWUpdate:Update..... 86
H commands..... 87
HDR..... 87
HEADer..... 87
HELPevery:ACQuire..... 87
HELPevery:ALL..... 88
HELPevery:CURsor..... 88
HELPevery:FFT..... 89
HELPevery:MATH..... 89
HELPevery:MEASUrement..... 90
HELPevery:REFerence..... 90
HELPevery:TRIGger..... 90
HELPevery:UTIlity..... 91
HELPevery:VERtical..... 91
HORizontal?..... 92
HORizontal:ACQLENGTH..... 92
HORizontal:DIVisions..... 93
HORizontal[:MAIn][:DELay]:POSition..... 93
HORizontal:MAIn:DELay:MODe..... 94
HORizontal:MAIn:DELay:STATe..... 94
HORizontal[:MAIn]:DELay:TIMe..... 95
HORizontal[:MAIn]:SAMPLERate..... 95
HORizontal[:MAIn]:SCAle..... 96
HORizontal[:MAIn]:SECdiv..... 96
HORizontal:MAIn:UNIts[:STRing]..... 97
HORizontal:PREViewstate..... 97
HORizontal:RECOrdlength..... 98
HORizontal:RECOrdlength:Auto..... 98
HORizontal:RESOlution..... 98
HORizontal:ROLL..... 99
HORizontal:TRIGger:POSition..... 99
I commands..... 100
ID?..... 100
*IDN?..... 100
L commands..... 102
LANGuage..... 102
LOCk..... 102
*LRN?..... 103
M commands..... 104
MATH?..... 104
MATH:DEFINE..... 104
MATH:HORizontal:POSition..... 105
MATH:HORizontal:SCALe..... 105
MATH:HORizontal:UNIts..... 106
MATH:LABel..... 106

MATH:VERtical:POSition ... 107
MATH:VERtical:SCAle ... 107
MATH:VERTical:UNIts ... 108
MEASUrement? ... 108
MEASUrement:CLEARSNapshot ... 109
MEASUrement:ENABLE ... 109
MEASUrement:GATing ... 110
MEASUrement:IMMed? ... 110
MEASUrement:IMMed:DELay? ... 111
MEASUrement:IMMed:DELay:EDGE<x> ... 111
MEASUrement:IMMed:SOUrce1 ... 111
MEASUrement:IMMed:SOUrce2 ... 112
MEASUrement:IMMed:TYPe ... 113
MEASUrement:IMMed:UNIts? ... 114
MEASUrement:IMMed:VALue? ... 115
MEASUrement:MEAS<x>? ... 115
MEASUrement:MEAS<x>:DELay? ... 116
MEASUrement:MEAS<x>:DELay:EDGE<x> ... 116
MEASUrement:MEAS<x>:SOUrce1 ... 117
MEASUrement:MEAS<x>:SOUrce2 ... 117
MEASUrement:MEAS<x>:STATE ... 118
MEASUrement:MEAS<x>:TYPe ... 118
MEASUrement:MEAS<x>:UNIts? ... 120
MEASUrement:MEAS<x>:VALue? ... 121
MEASUrement:REFLevel? ... 122
MEASUrement:REFLevel:ABSolute:LOW ... 122
MEASUrement:REFLevel:ABSolute:MID1 ... 123
MEASUrement:REFLevel:ABSolute:MID2 ... 123
MEASUrement:REFLevel:METHod ... 124
MEASUrement:REFLevel:PERCent:HIGH ... 124
MEASUrement:REFLevel:PERCent:LOW ... 125
MEASUrement:REFLevel:PERCent:MID1 ... 126
MEASUrement:REFLevel:PERCent:MID2 ... 126
MEASUrement:SNAPSHOT ... 127
MEASUrement:SOURCESNAPShot ... 127
O commands ... 128
*OPC ... 128
P commands ... 129
*PSC ... 129
R commands ... 130
*RCL ... 130
RECAll:SETUp ... 130
RECAll:WAVEForm ... 131
REF<x>? ... 131
REF<x>:DATE? ... 132
REF<x>:TIMe? ... 132
REF<x>:HORizontal:DELay:TIMe? ... 132
REF<x>:HORizontal:SCAle? ... 133
REF<x>:POSition? ... 133

REF<x>:VERTical:POSition? 133
REF<x>:VERTical:SCAle? 134
*RST 134
S commands 135
*SAV 135
SAVe:ASSIgn:TYPe 135
SAVe:IMAge 136
SAVe:IMAge:FILEFormat 136
SAVe:IMAGe:LAYout 137
SAVe:SETUp 137
SAVe:WAVEform 138
SAVe:WAVEform:FILEFormat 139
SELect:CH<x> 139
SELect:CONTROl 140
SELect:FFT 140
SELect:MATH 141
SELect:REF<x> 141
SET? 142
SETUP<x>:DATE? 142
SETUP<x>:TIME? (Query Only) 143
*SRE 143
*STB? 144
T commands 145
TEKSecure 145
TIMe 145
TRIGger 146
TRIGger:A 146
TRIGger:A:EDGE? 147
TRIGger:A:EDGE:COUPling 147
TRIGger:A:EDGE:SLOpe 148
TRIGger:A:EDGE:SOUrce 148
TRIGger:A:HOLDOff? 149
TRIGger:A:HOLDOff:TIMe 149
TRIGger:A:LEVel 150
TRIGger:A:LEVel:CH<x> 150
TRIGger:A:LOWerthreshold:CH<x> 151
TRIGger:A:MODe 152
TRIGger:A:PULse? 152
TRIGger:A:PULse:CLAss 153
TRIGger:A:PULSE:Width? 153
TRIGger:A:PULse:WIDth:POLarity 154
TRIGger:A:PULSEWidth:SOUrce 154
TRIGger:A:PULse:WIDth:WHEN 155
TRIGger:A:PULse:WIDth:WIDth 155
TRIGger:A:RUNT? 156
TRIGger:A:RUNT:POLarity 156
TRIGger:A:RUNT:SOUrce 157
TRIGger:A:RUNT:WHEn 157
TRIGger:A:RUNT:WIDth 158

TRIGger:A:TYPe....158
TRIGger:A:UPPerthreshold:CH<x>....159
TRIGger:FREQuency?....160
TRIGger:STATE?....160
U commands....161
UNLock....161
V commands....162
VERBose....162
W commands....163
*WAI....163
WAVFrm?....163
WFMInpre?....164
WFMInpre:BIT_Nr....164
WFMInpre:BYT_Nr....165
WFMInpre:ENCdg....165
WFMInpre:NR_Pt?....166
WFMInpre:XINcr....166
WFMInpre:XUNit....167
WFMInpre:XZEro....167
WFMInpre:YMUlt....168
WFMInpre:YOFf....168
WFMInpre:YUNit....169
WFMInpre:YZEro....170
WFMOutpre?....170
WFMOutpre:BIT_Nr....171
WFMOutpre:BN_Fmt....171
WFMOutpre:BYT_Nr....172
WFMOutpre:ENCdg....172
WFMOutpre:NR_Pt?....173
WFMOutpre:RECOrdlength?....173
WFMOutpre:WFId?....174
WFMOutpre:XINcr?....174
WFMOutpre:XUNit?....175
WFMOutpre:XZEro?....175
WFMOutpre:YMUlt?....176
WFMOutpre:YOFf?....176
WFMOutpre:YUNit?....177
WFMOutpre:YZEro?....177
Z commands....178
ZOOM?....178
ZOOm{:MODe|:STATE}....178
ZOOm:ZOOM1?....179
ZOOM:ZOOM1:FACtor....179
ZOOM:ZOOM1:HORizontal:POSition....179
ZOOM:ZOOM1:HORizontal:SCAle....180
ZOOm:ZOOM1:POSition....180
ZOOm:ZOOM1:SCAle....181
ZOOM:ZOOM1:STATE....181
Status and Events....183

Registers....183
Overview....183
Status Registers....183
Enable Registers....185
*PSC Command....185
Queues....186
Output Queue....186
Event Queue....186
Event Handling Sequence....186
Synchronization Methods....187
Overview....187
Using the *WAI Command....189
Using the BUSY Query....190
Using the *OPC Command....190
Using the *OPC? Query....192
Messages....193
No Event....193
Command Error....193
Execution Error....194
Device Error....198
System Event....198
Execution Warning....199
Internal Warning....200
Programming Examples....201
ASCII Code Chart....203
Factory setup....204
TBS1000C Series Oscilloscopes....204
Reserved words....205
Glossary terms....206
Index....207

List of Figures

Figure 1: Command message elements..... 15
Figure 2: Block argument example..... 21
Figure 3: The Standard Event Status Register (SESR)..... 183
Figure 4: The Status Byte Register (SBR)..... 184
Figure 5: The Device Event Status Enable Register (DESER)..... 185
Figure 6: The Event Status Enable Register (ESER)..... 185
Figure 7: The Service Request Enable Register (SRER)..... 185
Figure 8: Status and Event Handling Process..... 187
Figure 9: Command processing without using synchronization..... 189
Figure 10: Processing sequence with synchronization..... 189

List of Tables

Table 1: Instrument communication protocol....14
Table 2: BNF notation....14
Table 3: Command message elements....15
Table 4: Comparison of Header Off and Header On responses....16
Table 5: Types of numeric arguments....19
Table 6: Instrument handling of incorrect numeric arguments....19
Table 7: Parts of a block argument....20
Table 8: FPAnel:TURN arguments....86
Table 9: SESR bit functions....183
Table 10: SBR bit functions....184
Table 11: Instrument operations that can generate OPC....188
Table 12: No Event messages....193
Table 13: Command error messages (CME bit 5)....193
Table 14: Execution error messages (EXE bit 4)....194
Table 15: Device error messages (DDE bit 3)....198
Table 16: System event messages....199
Table 17: Execution warning messages (EXE bit 4)....199
Table 18: Execution warning messages (EXE bit 4)....200
Table 19: Internal warning messages....200

Getting Started

This manual contains information on how to remotely control and operate your instrument through communications protocol and commands.

Refer to the instrument user manual for information on how to configure and test your instrument remote connectivity (USB or Ethernet).

Download the latest version of the programmer manual from www.tek.com/downloads for up-to-date command syntax information.

Command Syntax

You can control the instrument through the Ethernet or USB interface using a large group of commands and queries.

This section describes the syntax these commands and queries use and the conventions the instrument uses to process them. The commands and queries themselves are listed in the *Command Descriptions* section.

Command Syntax

Table 1: Instrument communication protocol

Model or option	GPIB	RS-232	USB
TBS1000C	Yes	No	Yes

You transmit commands to the instrument using the enhanced American Standard Code for Information Interchange (ASCII) character encoding. *Appendix A* contains a chart of the ASCII character set.

The Backus Naur Form (BNF) notation is used in this manual to describe commands and queries.

Table 2: BNF notation

Symbol	Meaning
< >	Defined element
::=	Is defined as
|	Exclusive OR
{ }	Group; one element is required
[ ]	Optional; can be omitted
. . .	Previous element(s) may be repeated
( )	Comment

Command and Query Structure

Commands consist of set commands and query commands (usually simply called commands and queries). Commands change instrument settings or perform a specific action. Queries cause the instrument to return data and information about its status.

Most commands have both a set form and a query form. The query form of the command is the same as the set form except that it ends with a question mark. For example, the set command ACQuire:MODe has a query form ACQuire:MODe. Not all commands have both a set and a query form; some commands are set only and some are query only.

A few commands do both a set and query action. For example, the *CAL? command runs a self-calibration program on the instrument, then returns the result of the calibration.

A command message is a command or query name, followed by any information the instrument needs to execute the command or query. Command messages consist of five different element types.

Table 3: Command message elements

Symbol	Meaning
<Header>	The basic command name. If the header ends with a question mark, the command is a query. The header may begin with a colon (:) character; if the command is concatenated with other commands the beginning colon is required. The beginning colon can never be used with command headers beginning with a star (*).
<Mnemonic>	A header subfunction. Some command headers have only one mnemonic. If a command header has multiple mnemonics, they are always separated from each other by a colon (:) character.
<Argument>	A quantity, quality, restriction, or limit associated with the header. Not all commands have an argument, while other commands have multiple arguments. Arguments are separated from the header by a <Space>. Arguments are separated from each other by a <Comma>.
<Comma>	A single comma between arguments of multiple-argument commands. It may optionally have white space characters before and after the comma.
<Space>	A white space character between command header and argument. It may optionally consist of multiple white space characters.

The following figure shows the five command message elements.

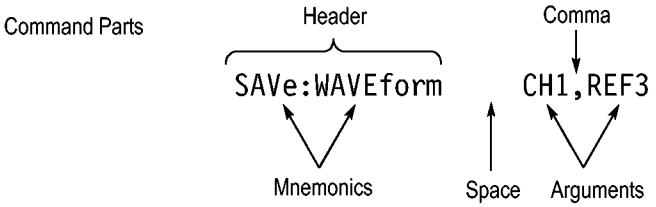


Figure 1: Command message elements

Commands

Commands cause the instrument to perform a specific function or change one of its settings. Commands have the structure:

[:]<Header>[<Space><Argument>[<Comma><Argument>]...]

A command header is made up of one or more mnemonics arranged in a hierarchical or tree structure. The first mnemonic is the base or root of the tree and each subsequent mnemonic is a level or branch off of the previous one. Commands at a higher level in the tree may affect those at a lower level. The leading colon (:) always returns you to the base of the command tree.

Queries

Queries cause the instrument to return information about its status or settings. Queries have the structure:

[:]<Header>

[:]<Header>[<Space><Argument>[<Comma><Argument>]...]

You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level.

For example, MEASUrement:MEAS<x>:UNIts? returns the measurement units, while MEASUrement:MEAS<x>:TYPe? returns the measurement type selected for the measurement, and MEASUrement:MEAS<x>? returns all the measurement parameters for the specified measurement.

Headers in Query Responses

You can control whether the instrument returns headers as part of the query response. Use the HEADer command to control this feature. If header is on, the instrument returns command headers as part of the query and formats the query response as a valid set command. When header is off, the instrument sends back only the values in the response. This format can make it easier to parse and extract the information from the response.

Table 4: Comparison of Header Off and Header On responses

Query	Header Off response	Header On response
ACQuire:NUMAVg	64	ACQUIRE:NUMAVG 64
CHx1:COUPling	DC	CH1:COUPLING DC

Clearing the Output Queue

To clear the output queue and reset the instrument to accept a new command or query, send a Device Clear (DCL) from a GPIB host.

From an RS-232 host, send a break signal. The RS-232 interface responds by returning the ASCII string "DCL."

From a USB host, send an INITIATE_CLEAR followed by a CHECK_CLEAR_STATUS. The USB interface responds to CHECK_CLEAR_STATUS with STATUS_SUCCESS when it is finished clearing the output queue.

Command Entry

Follow these general rules when entering commands:

- Enter commands in upper or lower case.
- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 decimal).
- The instrument ignores commands that consists of just a combination of white space characters and line feeds.

Abbreviating Commands

You can abbreviate many instrument commands. These abbreviations are shown in capital letters in the command listing in the *Command Groups* section and *Command Descriptions* section. For example, the command ACQuire:NUMAvg can be entered simply as ACQ:NUMA or acq:numa.

If you use the HEADer command to have command headers included as part of query responses, you can also control whether the returned headers are abbreviated or are full-length using the VERBose command.

Concatenating Commands

You can concatenate any combination of set commands and queries using a semicolon (;). The instrument executes concatenated commands in the order received. When concatenating commands and queries you must follow these rules:

- Completely different headers must be separated by both a semicolon and by the beginning colon on all commands but the first. For example, the commands TRIGger:MODe NORMal and ACQuire:NUMAVg 16 can be concatenated into a single command:

  TRIGger:MODe NORMal;:ACQuire:NUMAVg 16
- If concatenated commands have headers that differ by only the last mnemonic, you can abbreviate the second command and eliminate the beginning colon. For example, the commands ACQuire:MODe AVErage and ACQuire:NUMAVg 16 could be concatenated into a single command:

  ACQuire:MODe AVErage; NUMAVg 16

  The longer version works equally well:

  ACQuire:MODe AVErage;:ACQuire:NUMAVg 16
- Never precede a star (*) command with a colon or semicolon:

  ACQuire:MODe AVErage;*TRG

  The instrument processes commands that follow the star command as if the star command was not there, so:

  ACQuire:MODe AVErage;*TRG;NUMAVg 16

  sets the acquisition mode to average and sets acquisition averaging to 16. The *TRG command is ignored.
- When you concatenate queries, the responses to all queries are combined into a single response message. For example, if channel 1 coupling is set to DC and the bandwidth is set to 20 MHz, the concatenated query:

  CH1:COUPling;BANdwidth

  returns CH1:COUPLING DC;:CH1:BANDWIDTH ON if header is on, or DC;ON if header is off.
- You can concatenate set commands and queries in the same message. For example:

  ACQuire:MODe AVErage;NUMAVg;STATE

  is a valid message that sets the acquisition mode to average, queries the number of acquisitions for averaging, and then queries the acquisition state. The instrument executes concatenated commands and queries in the order it receives them.
- Any query that returns arbitrary data, such as ID, must be the last query when part of a concatenated command. If the query is not last, the instrument generates event message 440.

  Here are some INVALID concatenation examples:
- CH1:COUPling DC;ACQuire:NUMAVg 16 (missing colon before ACQuire)
- CH1:COUPling DC;:BANDwidth ON (invalid colon before BANDwidth)
- CH1:COUPling DC;:*TRG (invalid colon before a star (*) command)
- HORizontal:MAIn:POSition 0;MAIn:SCAle 1E-13 (levels of mnemonics are different; either remove the second occurrence of MAIn:, or put HORizontal: in front of MAIN:SCAle)

Message Terminators

This manual uses the term <EOM> (End of message) to represent a message terminator.

GPIB End of Message (EOM) Terminators

GPIB EOM terminators can be the END message (EOI asserted concurrently with the last data byte), the ASCII code for line feed (LF) sent as the last data byte, or both. The instrument always terminates messages with LF and EOI. White space is allowed before the terminator; for example, CR LF is acceptable.

USB End of Message (EOM) Terminators

The EOM bit must be set in the USB header of the last transfer of a command message

See the USB Test and Measurement Class Specification (USBTMC) section 3.2.1 for details. The instrument terminates messages by setting the EOM bit in the USB header of the last transfer of a message to the host (USBTMC Specification section 3.3.1), and by terminating messages with a LF. White space is allowed before the terminator; for example, CR LF is acceptable.

Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic could be CH2. You can use these mnemonics in the command just as you do any other mnemonic. For example, there is a CH1:VOLts command and there is also a CH2:VOLts command. In the command descriptions, this list of choices is abbreviated CH<x>.

Channel mnemonics

Commands specify the channel to use as a mnemonic in the header.

Symbol	Meaning
CH<x>	A channel specifier; <x> is 1 or 2.

Reference Waveform Mnemonics

Commands can specify the reference waveform to use as a mnemonic in the header.

Symbol	Meaning
REF<x>	A reference waveform specifier; <x> is 1 or 2.

Waveform Mnemonics

In some commands you can specify a waveform without regard to its type: channel waveform, math waveform, or reference waveform. The "y" is the same as "x" in Reference Waveform Mnemonics.

Symbol	Meaning
<wfm>	Can be CH<x>, MATH, or REF<y>

Cursor Position Mnemonic

When the instrument displays cursors, commands may specify which cursor of the pair to use.

Symbol	Meaning
POSITION<x>	A cursor selector;<x> is 1 or 2.

Measurement Specifier Mnemonics

Commands can specify which measurement to set or query as a mnemonic in the header. The instrument can display up to six automated measurements.

Symbol	Meaning
MEAS<x>	A measurement specifier; <x> is 1-6.

Argument Types

A command argument can be in one of several forms. The individual descriptions of each command tell which argument types to use with that command.

Numeric Arguments

Many instrument commands require numeric arguments.

Table 5: Types of numeric arguments

Symbol	Meaning
<NR1>	Signed integer value
<NR2>	Floating point value without an exponent
<NR3>	Floating point value with an exponent

The syntax shown is the data format that the instrument returns in response to a query. This format is also the preferred format when sending a command to the instrument.

When you enter an incorrect numeric argument, the instrument automatically forces the numeric argument to a correct value.

Table 6: Instrument handling of incorrect numeric arguments

Argument value	Instrument response
Numeric argument is less than lowest correct value for that command	Sets the specified command to the lowest correct value and executes the command
Numeric argument is greater than the highest correct value for that command	Sets the specified command to the highest correct value and executes the command
Numeric value is between two correct values	Rounds the entered value to the nearest correct value and executes the command

Quoted String Arguments

Some commands accept or return data in the form of a quoted string, which is simply a group of ASCII characters enclosed by single quotes (') or double quotes ("). For example:

"this is a quoted string"

Symbol	Meaning
<QString>	Quoted string of ASCII text

Follow these rules when you use quoted strings:

1. A quoted string can include any character defined in the 7-bit ASCII character set. *ASCII Code Chart* on page 203.
2. Use the same type of quote character to open and close the string:

   "this is a valid string"
3. You can mix quotation marks within a string if you follow the previous rule:

   "this is an 'acceptable' string"
4. You can include a quote character within a string simply by repeating the quote. For example,

   "here is a "" mark"
5. Strings can have upper or lower case characters.
6. If you use a GPIB network, you cannot terminate a quoted string with the END message before the closing delimiter.
7. A carriage return or line feed embedded in a quoted string does not terminate the string, but is treated as just another character in the string.
8. The maximum length of a quoted string returned from a query is 1000 characters.

Here are some examples of invalid strings:

"Invalid string argument' (quotes are not of the same type)

"test<EOI>" (termination character is embedded in the string)

Block Arguments

Several instrument commands use a block argument form.

Table 7: Parts of a block argument

Symbol	Meaning
<NZDig>	A nonzero digit character, in the range 1-9 Specifies the number of <Dig> elements that follow
<Dig>	A digit character, in the range 0-9
<DChar>	A character with the hex equivalent of 00 through FF hexadecimal (0 through 255 decimal)
<Block>	A block of data bytes, defined as: <Block> := { #<NZDig><Dig>[<Dig>...][<DChar>...] | #0[<DChar>...]<terminator> }

The following figure shows an example of a block argument.

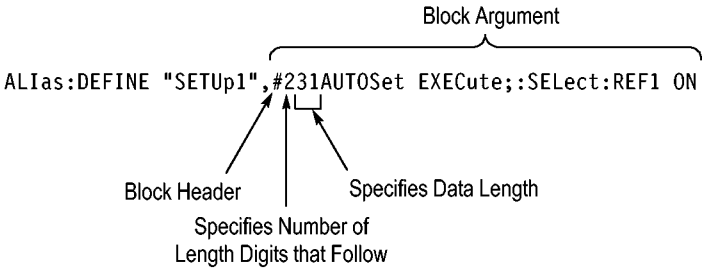


Figure 2: Block argument example

<NZDig> specifies the number of <Dig> elements that follow. Taken together, the <Dig> elements form a decimal integer that specifies how many <DChar> elements follow.

#0 means that the <Block> is an indefinite length block. The <terminator> ends the block. You should not use indefinite length blocks with RS-232, because there is no way to include a <terminator> character as a <DChar> character.

The first occurrence of a <terminator> character signals the end of the block and any subsequent <DChar> characters will be interpreted as a syntax error. With the GPIB, the EOI line signals the last byte. With the USB, the EOM bit signals the last byte.

A commands

This section lists commands and queries that begin with the letter A.

ACQuire?

Returns the current acquisition settings. Query only.

Group

Acquisition

Syntax

```
ACQuire?
```

Related Commands

ACQuire:MODe on page 22, *ACQuire:NUMACq?* on page 23, *ACQuire:NUMAVg* on page 24, *ACQuire:NUMAVg* on page 24

Returns

Returns current acquisition settings: Stop after, Acquisition state, Mode, Number of averages.

Examples

`ACQuire?` might return the following string for the current acquisition: `ACQUIRE:STOPAFTER RUNSTOP;STATE 1; MODE SAMPLE;NUMAVG 16`

ACQuire:MAXSamplerate?

Returns the maximum real-time sample rate, which varies from model to model. Query only.

Group

Acquisition

Syntax

```
ACQuire:MAXSamplerate?
```

Examples

`ACQUIRE:MAXSAMPLERATE?` might return `1.0000E+9` indicating the maximum real-time sample rate is 1.0 GS/s.

ACQuire:MODe

Sets or queries the acquisition mode of the instrument for all live waveforms.

Waveforms are the displayed data point values taken from acquisition intervals. Each acquisition interval represents a time duration set by the horizontal scale (time per division).

The instrument sampling system always samples at the maximum rate, so the acquisition interval may include more than one sample. The acquisition mode, which you set using this ACQuire:MODe command, determines how the final value of the acquisition interval is generated from the many data samples.

Group

Acquisition

Syntax

`ACQuire:MODe {SAMple|PEAKdetect|HIRes|AVErage}`

`ACQuire:MODe?`

Related commands

ACQuire:NUMAVg on page 24, *CURVe* on page 55

Arguments

`SAMple` specifies that the displayed data point value is the first sampled value that was taken during the acquisition interval. The waveform data has 8 bits of precision in all acquisition modes. You can request 16 bit data with a CURVe? query, but the lower-order 8 bits of data will be zero. `SAMple` is the default mode.

`PEAKdetect` specifies the display of the high-low range of the samples taken from a single waveform acquisition. The instrument displays the high-low range as a vertical column that extends from the highest to the lowest value sampled during the acquisition interval. PEAKdetect mode can reveal the presence of aliasing or narrow spikes.

`HIRes` specifies Hi Res mode where the displayed data point value is the average of all the samples taken during the acquisition interval. This is a form of averaging, where the average comes from a single waveform acquisition. The number of samples taken during the acquisition interval determines the number of data values that compose the average.

`AVErage` specifies averaging mode, in which the resulting waveform shows an average of `SAMple` data points from several separate waveform acquisitions. The instrument processes the number of waveforms you specify into the acquired waveform, creating a running exponential average of the input signal. The number of waveform acquisitions that go into making up the average waveform is set or queried using the ACQuire:NUMAVg command.

Examples

`ACQuire:MODe AVErage` sets average acquisition mode so that the resulting waveform is the average of the specified number of waveform acquisitions.

`ACQuire:MODe?` might return `ACQUIRE:MODE AVERAGE` indicating that the displayed waveform is the average of the specified number of waveform acquisitions.

ACQuire:NUMACq?

Indicates the number of acquisitions that have taken place since starting instrument acquisition. The acquisition number will continue to increase while acquisitions are running until there is a reset.

Starting and stopping acquisitions do not cause this number to reset. For example, if acquisitions are running, the acquisition count will increase (assuming the instrument is triggering). If you stop the acquisitions, the acquisition number will freeze at a given number (For example: 5000). If you start acquisitions again, it will continue from 5000. The number will reset to 0 only if you change the horizontal scale while acquisitions are running.

Group

Acquisition

Syntax

`ACQuire:NUMACq?`

Related commands

ACQuire:STATE on page 24

Returns

`<NR1>`

Examples

`ACQuire:NUMACq?` might return `ACQUIRE:NUMACQ 350` indicating that 350 acquisitions have occurred.

ACQuire:NUMAVg

Sets or queries the number of waveform acquisitions that make up an averaged waveform. Use the `ACQuire:MODe` command to enable Average mode. Sending this command is equivalent to turning a multipurpose knob to enter the number of waveform acquisitions to average.

Group

Acquisition

Syntax

`ACQuire:NUMAVg <NR1>`

`ACQuire:NUMAVg?`

Arguments

`<NR1>` is the number of waveform acquisitions to average. The range of values is from 2 to 512 in powers of two.

Examples

`ACQuire:NUMAVg 16` specifies that 16 waveform averages are performed before exponential averaging starts.

`ACQuire:NUMAVg?` might return `ACQUIRE:NUMAVG 64` indicating that there are 64 acquisitions specified for averaging.

ACQuire:STATE

Starts or stops acquisitions.

When State is set to ON or RUN, a new acquisition is started. If the last acquisition was a single acquisition sequence, a new single sequence acquisition is started. If the last acquisition was continuous, a new continuous acquisition is started.

If RUN is issued in the middle of completing a single sequence acquisition (for example, averaging or enveloping), the acquisition sequence is restarted, and any accumulated data is discarded. Also, the instrument resets the number of acquisitions. If the RUN argument is issued while in continuous mode, acquisition continues.

Group

Acquisition

Syntax

`ACQuire:STATE {OFF|ON|RUN|STOP|<NR1>}`

`ACQuire:STATE?`

Related Commands

**OPC* on page 128, *ACQuire:STOPAfter* on page 25

Arguments

`OFF|STOP|<NR1>` = 0 stops acquisitions; any other value starts acquisitions..

`ON|RUN|<NR1>` ≠ 0 starts acquisition and display of waveforms.

Examples

`ACQuire:STATE RUN` starts acquisition of waveform data and resets the number of acquisitions count (NUMACq) to zero.

`ACQuire:STATE?` might return: `ACQUIRE:STATE 0` indicating that the acquisition is stopped.

ACQuire:STOPAfter

Sets or returns whether the instrument continually acquires acquisitions or acquires a single sequence.

Group

Acquisition

Syntax

`ACQuire:STOPAfter {RUNSTop|SEQuence}`

`ACQuire:STOPAfter?`

Related commands

ACQuire:STATE on page 24

Arguments

`RUNSTop` specifies that the instrument will continually acquire data, if `ACQuire:STATE` is turned on.

`SEQuence` specifies that the next acquisition will be a single-sequence acquisition.

Examples

`ACQuire:STOPAfter` `RUNSTOP sets the instrument to continually acquire data.

`ACQuire:STOPAfter`? might return: ACQUIRE:STOPAFTER SEQUENCE indicating that the next acquisition the instrument makes will be of the single-sequence type.

ALIas

Sets or queries the state of alias functionality.

Use Alias commands to define new commands as a sequence of standard commands. You may find this useful when repeatedly using the same commands to perform certain tasks like setting up measurements. Aliases are similar to macros but do not include the capability to substitute parameters into alias bodies.

To use Alias commands, first define the alias, then turn on the alias state.

Group

Alias

Syntax

```
ALIas {OFF|ON|<NR1>}
ALIas?
```

Related commands

ALIas:DEFine on page 27, *ALIas[:STATE]* on page 29

Arguments

`OFF` turns alias expansion off. If a defined alias is sent when ALIas is off, a command error (110) will be generated.

`ON` turns alias expansion on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

`<NR1> = 0` disables alias mode; any other value enables alias mode.

Examples

`ALIAS ON` turns the alias feature on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

`ALIAS?` might return `:ALIAS 1` indicating that the alias feature is on.

ALIas:CATalog?

Returns a list of the currently defined alias labels, separated by commas. If no aliases are defined, the query returns the string "". Query only.

Group

Alias

Syntax

```
ALIas:CATalog?
```

Examples

`ALIAS:CATALOG?` might return the string `:ALIAS:CATALOG "SETUP1","TESTMENU1","DEFAULT"` showing that there are three aliases named SETUP1, TESTMENU1, and DEFAULT.

ALIas:DEFine

Assigns a sequence of program messages to an alias label. These messages are then substituted for the alias whenever it is received as a command or query, provided that ALIas:STATE has been turned on. The query form of this command returns the definitions of a selected alias.

Note: Attempting to give two aliases the same name causes an error. To give a new alias the name of an existing alias, the existing alias must first be deleted.

Group

Alias

Syntax

```
ALIas:DEFine <QString><,>{<QString>|<Block>}
```

```
ALIas:DEFine? <QString>
```

Related commands

ALIas[:STATE] on page 29

Arguments

The first <QString> is the alias label. This label cannot be a command name. Labels must start with a letter and can contain only letters, numbers, and underscores; other characters are not allowed. The label must be less than or equal to 12 characters. The second<QString> or <Block> is a complete sequence of program messages. The messages can contain only valid commands that must be separated by semicolons and must follow all rules for concatenating commands. The sequence must be less than or equal to 256 characters.

Examples

`ALIAS:DEFINE "ST1",":RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON"` defines an alias named `"ST1"` that sets up the instrument.

`ALIAS:DEFINE? "ST1"` returns `:ALIAS:DEFINE "ST1",#246 :RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON`.

ALIas:DELEte

Removes a specified alias and is identical to ALIas:DELEte:NAMe. An error message is generated if the named alias does not exist. (No query form.

Group

Alias

Syntax

```
ALIas:DELEte <QString>
```

Related commands

**ESR?* on page 70, *ALIas:DELEte:ALL* on page 28

Arguments

<QString> is the name of the alias to be removed. Using ALIas:DELEte without specifying an alias causes an execution error. <QString> must be a previously defined value.

Examples

ALIAS:DELETE "SETUP1" deletes the alias named SETUP1.

ALIas:DELEte:ALL

Deletes all existing aliases. No query form.

Group

Alias

Syntax

ALIas:DELEte:ALL

Related commands

ALIas:DELEte on page 27, *ALIas:DELEte[:NAMe]* on page 28

Examples

ALIAS:DELETE:ALL deletes all existing aliases.

ALIas:DELEte[:NAMe]

Removes a specified alias. This command is identical to ALIas:DELEte. No query form.

Group

Alias

Syntax

ALIas:DELEte[:NAMe] <QString>

Arguments

<QString> is the name of the alias to remove. Using ALIas:DELEte[:NAMe] without specifying an alias causes an execution error. <QString> must be an existing alias.

Examples

ALIAS:DELETE:NAME "STARTUP" deletes the alias named STARTUP.

ALIas[:STATE]

Turns aliases on or off. This command is identical to the ALIas command.

Group

Alias

Syntax

`ALIas[:STATE] {<NR1>|OFF|ON}`

`ALIas[:STATE]?`

Arguments

`OFF` or `<NR1> = 0` turns alias expansion off. If a defined alias is sent when `ALIas:STATE` is `OFF`, a command error (102) is generated.

`ON` or `<NR1> ≠ 0` turns alias expansion on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

Examples

`ALIAS:STATE OFF` turns the command alias feature off.

`ALIAS[:STATE]?` returns 0 when the alias feature is off.

ALLEv?

Causes the instrument to return all events and their messages, and removes the returned events from the Event Queue. The messages are separated by commas. Use the *ESR? query to enable the events to be returned. Refer to the Status and Events section for a complete description of how to use these registers. This command is similar to repeatedly sending *EVMsg? queries to the instrument. Query only.

Group

Status and Error

Syntax

`ALLEv?`

Related Commands

*CLS on page 46, DESE on page 61, *ESE on page 69, *ESR? on page 70, EVENT? on page 70, EVMsg? on page 71, EVQty? on page 72, *SRE on page 143, *STB? on page 144

Returns

The event code and message in the following format:

`<Event Code><Comma><QString>[<Comma><Event Code><Comma><QString>...]`

<QString>::= <Message>;[<Command>]

<Command> is the command that caused the error and may be returned when a command error is detected by the instrument. As much of the command is returned as possible without exceeding the 60 character limit of the <Message> and <Command> strings combined. The command string is right-justified.

Examples

`ALLEv?` might return the following string: `ALLEV 2225,"MEASUREMENT ERROR, NO WAVEFORM TO MEASURE; ",420,"QUERY UNTERMINATED; "`

AUTOSet

Causes the instrument to adjust its vertical, horizontal, and trigger controls to display a stable waveform. This command is equivalent to pushing the front-panel AUTOSET button. For a detailed description of the Autoset function, refer to the user manual for your instrument. Command only, no query form.

Group

Miscellaneous

Syntax

`AUTOSet {EXECute | UNDo}`

Arguments

`EXECute` runs Autoset on the selected waveform.

`UNDo` restores the oscilloscope settings to those prior to running Autoset.

AUTOSet:ENABLE

Allows educators to disable or enable the Autorange and Autoset functions. The function can be manually set from the Utility menu. To access the menu, refer to the product user manual.

Group

Miscellaneous

Syntax

`AUTOSet:ENABLE {ON | OFF}`

`AUTOSet:ENABLE?`

Related commands

AUTOSet on page 30

FEAEN:PASSWORD on page 73

Arguments

`ON` enables the autoset feature.

`OFF` disables the autoset feature.

Examples

`AUTOSET:ENABLE OFF` disables autoset.

`AUTOSET:ENABLE?` might return `1` indicating that autoset is enabled.

B commands

This section lists commands and queries that begin with the letter B.

BUSY?

Returns the status of the instrument. This command allows you to synchronize the operation of the instrument with your application program. Query only.

Certain instrument operations can affect the BUSY? response. *Table 11* on page 188

Group

Status and Error

Syntax

`BUSY?`

Related Commands

**OPC* on page 128, **WAI* on page 163

Returns

`<NR1>` = 0 means the instrument is not busy processing a command whose execution time is extensive.

`<NR1>` = 1 means the instrument is busy processing a command whose execution time is extensive. *Table 11* on page 188

Examples

`BUSY?` might return `:BUSY 1` indicating that the instrument is now busy. See *Using the BUSY Query* on page 190 for an example of how to use this query.

C commands

This section lists commands and queries that begin with the letter C.

*CAL?

Performs an internal self-calibration and returns its status. This is equivalent to selecting the Do Self Cal option in the Utility menu. Although *CAL? is a query command, it does perform an action. Query only.

Note: The self-calibration can take several minutes to complete. During this time, the instrument does not execute any commands.

Disconnect all signals from the instrument before performing an internal self-calibration.

Group

Calibration and Diagnostic

Syntax

`*CAL?`

Related Commands

CALibrate:INTERNAL on page 33

Returns

`0` indicates that the self-calibration completed without any errors detected.

Any value other than zero indicates that the self-calibration did not complete successfully or completed with errors.

Examples

`*CAL?` performs a self-calibration and might return 0 to indicate that it completed successfully.

CALibrate:INTERNAL

This command starts a signal path compensation. Command only, no query form.

Note: The self-calibration can take several minutes to complete. During this time, the instrument does not execute any commands.

Disconnect all signals from the instrument before performing an internal self-calibration.

Group

Calibration and Diagnostic

Syntax

`CALibrate:INTERNAL`

Examples

`CALibrate:INTERNAL` starts a signal path compensation cycle.

CALibrate:INTERNal:STARt

Starts the internal signal path calibration (SPC) of the instrument. You can use the `CALibrate:INTERNal:STATus?` query to return the current status of the internal signal path calibration of the instrument. No query form.

Group

Calibration and Diagnostic

Syntax

`CALibrate:INTERNal:STARt`

Related commands

CALibrate:RESults:SPC? on page 35

Examples

`CALIBRATE:INTERNAL:START` initiates the internal signal path compensation of the instrument.

CALibrate:INTERNal:STATus?

Returns the current status of the instrument internal signal path compensation for the last SPC operation. Query only.

Group

Calibration and Diagnostic

Syntax

`CALibrate:INTERNal:STATus?`

Related commands

**CAL?* on page 33

Returns

`INIT` indicates the instrument has not had internal signal path calibration run.

`PASS` indicates the signal path calibration completed successfully.

`FAIL` indicates the signal path calibration did not complete successfully.

`RUNNING` indicates the signal path calibration is currently running.

Examples

`CALibrate:INTERNal:STATus?` might return `:CALIBRATE:INTERNAL:STATUS INIT` indicating that the current status of the internal signal path compensation is that it has not been run.

CALibrate:RESults?

Returns the status of internal and factory calibrations, without performing any calibration operations. Query only.

The results returned do not include the calibration status of attached probes. The CALibrate:RESults? query is intended to support GO/NoGO testing of the instrument calibration readiness: all returned results should indicate PASS status if the instrument is fit for duty. It is quite common, however, to use uncalibrated probes (particularly when the instrument inputs are connected into a test system with coaxial cables).

Group

Calibration and Diagnostic

Syntax

CALibrate:RESults?

Related commands

*CAL? on page 33

Examples

CALibrate:RESults? might return :CALibrate:RESults INIT indicating the instrument has not be calibrated.

CALibrate:RESults:SPC?

Returns the status of the SPC operation. This query does not initiate a SPC. Query only.

Group

Calibration and Diagnostic

Syntax

CALibrate:RESults:SPC?

Related commands

*CAL? on page 33

Returns

INIT indicates that SPC has never successfully completed.

PASS indicates that the last SPC operation passed.

FAIL indicates that the last SPC operation failed.

RUNNING indicates that the SPC operation is running.

Examples

CALibrate:RESults:SPC? might return :CALibrate:RESults:SPC INIT indicating SPC has not be run successfully.

CH<x>?

Returns the vertical parameters for the specified channel. The value of <x> can vary from 1 through 4 depending on instrument model. Query only.

Because CH<x>:SCAle and CH<x>:VOLts are identical, only CH<x>:SCAle is returned.

Group

Vertical

Syntax

`CH<x>?`

Related Commands

SELect:REF<x> on page 141

Returns

instrument vertical settings for the specified channel.

Examples

`CH1?` might return `:CH1:SCALE 1.0E0;POSITION 0.0E0; COUPLING DC;BANDWIDTH FULL;PROBE 1.0E0.`

CH<x>:AMPSVIAVOLTs:ENABLe

Sets or queries measure current status as ON or Off. The value <x> can vary from 1 through 2 depending upon the channel.

Group

Vertical

Syntax

`CH<x>:AMPSVIAVOLTs:ENABLe {ON|OFF|<NR>}`

`CH<x>:AMPSVIAVOLTs:ENABLe?`

Arguments

OFF turns current status off.

ON turns current status on.

<NR1> = 0 turns current status off; any other value turns current status on.

Examples

`Ch1:AMPSVIAVOLTS:ENABLE ON` will change the Ch1 measure current status as Yes.

CH<x>:AMPSVIAVOLTs :Factor

Sets or queries current factor . The value <x> can vary from 1 through 2 depending upon the channel .

Group

Vertical

Syntax

`CH<x>:AMPSVIAVOLTs:FACTOR <NR3>`

`CH<x>:AMPSVIAVOLTs:FACTOR?`

Arguments

`<NR3>` is the factor value.

Examples

`CH<x>:AMPSVIAVOLTs:FACTOR 1` will set it as 1 A per volt measurement.

`Ch1:AMPSVIAVOLTS:ENABLE ON` will change the Ch1 measure current status as Yes.

CH<x>:BANdwidth

Sets or queries the selectable low-pass bandwidth limit filter setting of the specified instrument channel. The value of <x> can vary from 1 through 2 depending on instrument model.

This command is equivalent to setting the BW Limit option in the Vertical menu.

Group

Vertical

Syntax

`CH<x>:BANdwidth {TWEnty|FULl|<NR3>}`

`CH<x>:BANdwidth?`

Arguments

`TWEnty` sets the upper bandwidth limit of channel <x> to 20 MHz.

`FULl` disables any optional bandwidth limiting. The specified channel operates at its maximum attainable bandwidth.

`<NR3>` is a double-precision ASCII string. The instrument rounds this value to an available bandwidth using geometric rounding, and then uses this value to set the upper bandwidth limit.

NOTE. *Other values may be possible depending on the attached probes.*

Examples

`CH1:BANDWIDTH TWENTY` sets the bandwidth of channel 1 to 20 MHz.

`CH1:BANDWIDTH?` might return `FULl`. This indicates there is no bandwidth limiting on channel 1.

CH<x>:COUPling

Sets or queries the input attenuator coupling setting of the specified instrument channel. The value of <x> can vary from 1 through 2 depending on the instrument model.

This command is equivalent to setting the Coupling option in the Vertical menu.

Group

Vertical

Syntax

`CH<x>:COUPling {AC | DC}`

`CH<x>:COUPling?`

Arguments

`AC` sets the specified instrument channel to AC coupling.

`DC` sets the specified instrument channel to DC coupling.

Examples

`CH1:COUPLING AC` establishes AC coupling on channel 1.

`CH2:COUPLING?` might return `:CH2:COUPling DC` indicating that channel 2 is set to DC coupling.

CH<x>:DESKew

Sets or queries the deskew time for channel <x>, where x is the channel number. You can adjust the deskew time to add an independent, channel-based delay time to the delay (set by the horizontal position control and common to all channels) from the common trigger point to first sample taken for each channel. This lets you compensate individual channels for different delays introduced by their individual input hook ups.

Group

Vertical

Syntax

`CH<x>:DESKew <NR3>`

`CH<x>:DESKew?`

Arguments

`<NR3>` is the deskew time for channel <x>, ranging from -100 ns to +100 ns with a resolution of 1 ns.

Examples

`CH1 :DESKew 5.0E-9` sets the deskew time for channel 1 to 5 ns.

`CH2:DESKew?` might return `:CH2:DESKEW 2.0000E-09` indicating that the deskew time for channel 2 is set to 2 ns.

CH<x>:INVert

Sets or queries the inversion state of the specified instrument channel. The value of <x> can vary from 1 through 2 depending on the instrument model.

This command is equivalent to setting the Invert option in the Vertical channel menus.

Group

Vertical

Syntax

`CH<x>:INVert {ON|OFF}`

`CH<x>:INVert?`

Arguments

`ON` inverts the specified instrument channel.

`OFF` sets the specified instrument channel to noninverted.

Examples

`CH1:INVERT ON` inverts the signal on channel 1.

`CH2:INVERT?` might return `:CH2:INVERT 0`, indicating that channel 2 is not inverted.

CH<x>:LABel

This command sets or queries the waveform label for channel<x>, where x is the channel number (1- 2).

Group

Vertical

Syntax

`CH<x>:LABel <Qstring>`

`CH<x>:LABel?`

Arguments

`<Qstring>` is an alphanumeric string of text, enclosed in quotes, that contains the text level information for the channel<x>waveform. The text string is limited to 30 characters.

Examples

`CH1:LABEL "ICCDATA"` sets the label name of Channel 1 waveform output to ICCDATA.

`CH1:LABEL?` might return `"ICCDATA"`, if the channel label was already set, else would return `""` if not set.

CH<x>:OFFSet

Sets or queries the vertical offset for channel <x>, where x is the channel number.

This command offsets the vertical acquisition window (moves the level at the vertical center of the acquisition window) for the specified channel. Visualize offset as scrolling the acquisition window towards the top of a large signal for increased offset values, and scrolling towards the bottom for decreased offset values. The resolution of the vertical window sets the offset increment for this control.

Offset adjusts only the vertical center of the acquisition window for channel waveforms to help determine what data is acquired. The instrument always displays the input signal minus the offset value.

The channel offset range depends on the vertical scale factor. The valid ranges for the instrument are (when the probe and external attenuation factor is X1):

For V/Div settings from 2 mV/div to 200 mV/div, the offset range is +/- 0.8 V

For V/Div settings from 202 mV/div to 5 V/div, the offset range is +/- 20 V

Group

Vertical

Syntax

`CH<x>:OFFSet <NR3>`

`CH<x>:OFFSet?`

Related commands

CH<x>:POSition on page 40

Arguments

`<NR3>` is the offset value for the specified channel <x>.

Examples

`CH1:OFFSet 2.0E-3` sets the offset for channel 1 to 2 mV.

`CH2:OFFSet?` might return `: CH2:OFFSET 1.0000E-03` indicating that the offset for channel 2 is set to 1 mV.

CH<x>:POSition

Sets or queries the vertical position of the specified instrument channel. The value of <x> can vary from 1 through 2 depending on the instrument model.

The position voltage value is applied to the signal before digitization. Increasing the position value of a waveform causes the waveform to move up. Decreasing the position value causes the waveform to move down. The position value determines the vertical graticule coordinate at which input signal values, minus the present offset setting for that channel, are displayed. For example, if the position for Channel 1 is set to 2.0 and the offset is set to 3.0, then input signals equal to 3.0 units are displayed 2.0 divisions above the center of the screen (at 1 V/div).

This command is equivalent to adjusting the front-panel VERTICAL POSITION knob.

Group

Vertical

Syntax

CH<x>:POSition <NR3>

CH<x>:POSition?

Related commands

CH<x>:OFFSet on page 40, *REF<x>:VERTical:POSition?* on page 133, *MATH:VERtical:POSition* on page 107

Arguments

<NR3> is the position in divisions from the center graticule for the specified channel. The range is 5 to -5 divisions.

Examples

CH2:POSITION 1.3E0 positions the channel 2 input signal 1.3 divisions above the center of the display.

CH1:POSITION? might return :CH1:POSITION -1.3000 indicating that the vertical position of Channel 1 is 1.3 divisions below the center graticule.

CH<x>:PRObe

Returns all information concerning the probe attached to channel <x>, where x is the channel number. The value of <x> can vary from 1 through 2 depending on the instrument model.

Group

Vertical

Syntax

CH<x>:PRObe?

Examples

CH1:PROBE? might return No probe.

CH<x>:PRObe:GAIN

Sets or queries the gain factor for the probe attached to the channel specified by <x>, where x is the channel number. The gain of a probe is the output divided by the input transfer ratio. For example, a common 10x probe has a gain of 0.1.

Group

Vertical

Syntax

CH<x>:PRObe:GAIN <NR3>

CH<x>:PRObe:GAIN?

Related commands

CH<x>:SCAle on page 44

Arguments

<NR3> is the probe gain. Allowed values depend on the specific probe.

Examples

CH1:PRObe:GAIN 0.1 sets the channel 1 probe gain to 0.1.

CH2:PROBE:GAIN? might return :CH2:PROBE:GAIN 0.1000E+00 indicating that the attached 10x probe delivers 1 V to the channel 2 BNC for every 10 V applied to the probe input.

CH<x>:PRObe:ID?

Returns the type and serial number of the probe attached to channel <x>, where x is the channel number. Query only.

Group

Vertical

Syntax

CH<x>:PRObe:ID?

Examples

CH2:PROBE:ID? might return :CH2:PROBE:ID:TYPE "10X";SERNUMBER "N/A" indicating that a passive 10x probe of unknown serial number is attached to channel 2.

CH<x>:PRObe:ID:SERnumber?

Returns the serial number of the probe attached to channel <x>, where x is the channel number. Query Only.

Note: For Level 0 and 1 probes, the serial number will be "".

Group

Vertical

Syntax

CH<x>:PRObe:ID:SERnumber?

Examples

CH1:PROBE:ID:SERNUMBER? might return :CH1:PROBE:ID:SERNUMBER "B010289" indicating that the serial number of the probe attached to channel 1 is B010289.

CH<x>:PRObe:ID:TYPE?

Returns the type of probe attached to the channel specified by <x>, where x is the channel number. Level 2 (or higher) probes supply their exact product nomenclature; for Level 0 or 1 probes, a generic “No Probe Detected message is returned. Query Only.

Group

Vertical

Syntax

`CH<x>:PRObe:ID:TYPE?`

Examples

`CH1:PROBE:ID:TYPE?` might return `:CH1:PROBE:ID:TYPE "P6203"` indicating that a P6203-type probe is attached to channel 1.

CH<x>:PRObe:SIGnal

Sets or queries the input bypass setting of a TekVPI probe attached to channel <x>, where x is the channel number. The probe must support input bypass, for example TCP0001. This command is ignored if sent to an unsupported probe.

Group

Vertical

Syntax

`CH<x>:PRObe:SIGnal {BYPass|PASS}`

`CH<x>:PRObe:SIGnal?`

Arguments

`BYPass` sets the probe to Bypass mode.

`PASS` sets the probe to Pass mode.

Examples

`CH1:PRObe:SIGnal PASS` set the probe attached to channel 1 to Pass mode.

`CH1:PRObe:SIGnal?` might return `:CH1:PRObe:SIGnal PASS` indicating that the probe attached to channel 1 is in Pass mode

CH<x>:PRObe:UNIts?

Returns a string describing the units of measure for the probe attached to channel <x>, where x is the channel number. Query Only.

Group

Vertical

Syntax

CH<x>:PRObe:UNIts?

Examples

CH2:PROBE:UNITS? might return :CH4:PROBE:UNITS "V" indicating that the units of measure for the probe attached to channel 2 are volts.

CH<x>:SCAle

Sets or queries the vertical scale of the specified instrument channel. The value of <x> can vary from 1 through 2 depending on the instrument model.

Each waveform has a vertical scale parameter. For a signal with constant amplitude, increasing the Scale causes the waveform to be displayed smaller. Decreasing the scale causes the waveform to be displayed larger.

Scale affects all waveforms, but affects channel waveforms differently from other waveforms:

For channel waveforms, this setting controls the vertical size of the acquisition window as well as the display scale. The range and resolution of scale values depends on the probe attached and any other external factors you have specified.

For reference and math waveforms, this setting controls the display only, graphically scaling these waveforms and having no affect on the acquisition hardware.

This command is equivalent to adjusting the front-panel VOLTS/DIV knob.

Group

Vertical

Syntax

CH<x>:SCAle <NR3>

CH<x>:SCAle?

Related Commands

CH<x>:OFFSet on page 40, *CH<x>:POSition* on page 40, *REF<x>:VERTical:SCAle?* on page 134, *MATH:VERtical:SCAle* on page 107

Arguments

<NR3> is the scale, in units-per-division. The value entered here is truncated to three significant digits.

Examples

CH1:SCALE 100E-3 sets the channel 1 gain to 100 mV/div.

CH2:SCALE? might return :CH2:SCALE 1.0000, indicating that the current V/div setting of channel 2 is 1 V/div.

CH<x>:VOLts

Sets or queries the vertical sensitivity of the specified channel. The value of <x> can vary from 1 through 2 depending on the instrument model.

This command is identical to the CH<x>:SCAle command and is included for compatibility purposes. Only CH<x>:SCAle is returned in response to a CH<x>? query.

Group

Vertical

Syntax

`CH<x>:VOLts <NR3>`

`CH<x>:VOLts?`

Arguments

`<NR3>` is the vertical sensitivity, in volts.

Examples

`CH1:VOLts 1.0` sets channel 1 to 1 Volt per division.

`CH1:VOLts?` Might return CH1:VOLts 1.0 indication that the ch1 volts per division is 1 Volt per division.

CH<x>:YUNit

Sets or queries the units of the specified channel.

String arguments are case insensitive and any unsupported units will generate an error. Supported units are:

%, /Hz, A, A/A, A/V, A/W, A/dB, A/s, AA, AW, AdB, As, B, Hz, IRE, S/s, V, V/A, V/V, V/W, V/dB, V/s, VV, VW, VdB, Volts, Vs, W, W/A, W/V, W/W, W/dB, W/s,WA, WV,WW, WdB, Ws, dB, dB/A, dB/V, dB/W, dB/dB, dBA, dBV, dBW, dBdB, day, degrees, div, hr, min, ohms, percent, s.

Group

Vertical

Syntax

`CH<x>:YUNit <QString>`

`CH<x>:YUNit?`

Arguments

`<QString>` is a string of text surrounded by quotes, specifying the supported units. This command is case insensitive.

Examples

`CH2:YUNit "V"` sets the units for channel 2 to Volts.

`CH2:YUNIT` might return `CH2:YUNIT "V"`, indicating that the channel 2 units are volts.

CLEARMenu

Clears the current menu from the display. This command is equivalent to pressing the front panel Menu off. No query form.

Group

Miscellaneous

Syntax

```
CLEARMenu
```

Examples

`CLEARMenu` clears the current menu from the display.

*CLS

Command only, no query form. The *CLS command clears the following instrument status data structures:

- The Event Queue
- The Standard Event Status Register (SESR)
- The Status Byte Register (except the MAV bit)

If the *CLS command immediately follows an <EOI>, the Output Queue and MAV bit (Status Byte Register bit 4) are also cleared. MAV indicates information is in the output queue. The device clear (DCL) GPIB control message and the USBTMC INITIATE_CLEAR control message will clear the output queue and also MAV.

*CLS does not clear the output queue or MAV. *CLS can suppress a service request that is to be generated by an *OPC command. This will happen if a hard copy output or single sequence acquisition operation is still being processed when the *CLS command is executed. See *Registers* on page 183

Group

Status and Error

Syntax

```
*CLS
```

Related Commands

DESE on page 61, **ESE* on page 69, **ESR?* on page 70, *EVENT?* on page 70, *EVMsg?* on page 71, **SRE* on page 143, **STB?* on page 144

Examples

`*CLS` clears the instrument status data structures.

CURSor?

Returns current cursor settings. Query only.

Group

Cursor

Syntax

`CURSor?`

Returns

instrument cursor settings.

Examples

`CURSor?` might return the following string as the current cursor settings: `:CURSOR:FUNCTION SCREEN;HBARS:POSITION1 0.0000;POSITION2 0.0000;UNITS BASE;:CURSOR:MODE INDEPENDENT;VBARS:POSITION1 -19.0006E-6;POSITION2 -18.9994E-6;UNITS SECONDS.`

CURSor:ENABLE

Allows educators to disable or enable the Cursor functions. The function can be manually set from the Utility menu. To access the menu, refer to the product user manual.

Group

Miscellaneous

Syntax

`CURSor:ENABLE {ON | OFF}`

`CURSor:ENABLE?`

Related commands

CURSor

FEAEN:PASSWORD on page 73

Arguments

`ON` enables the cursor feature.

`OFF` disables the cursor feature.

Examples

`CURSOR:ENABLE OFF` disables cursor.

`CURSOR:ENABLE?` might return `1` indicating that cursor is enabled.

CURSor:FUNCtion

Sets or queries the instrument cursor type. Cursors are attached to the selected waveform in Waveform mode and are attached to the display area in Screen mode.

Group

Cursor

Syntax

`CURSor:FUNCtion {OFF|SCREEN|TIME|AMPLitude}`

`CURSor:FUNCtion?`

Arguments

`OFF` removes the cursors from the display but does not change the cursor type.

`SCREEN` specifies both horizontal and vertical bar cursors, which measure the selected waveform in horizontal and vertical units. Use these cursors to measure anywhere in the waveform display area.

`TIME` specifies the vertical bar cursor to measure the selected waveform in vertical units.

`AMPLitude` specifies the horizontal bar cursor to measure the selected waveform in horizontal units.

Examples

`CURSor:FUNCtion TIME` selects the paired cursors for measuring waveform time.

`CURSor:FUNCtion?` might return `CURSOR:FUNCTION SCREEN` indicating that the screen cursors are currently selected.

CURSor:HBArs?

Returns the settings for the instrument horizontal bar cursors. Query only.

Group

Cursor

Syntax

`CURSor:HBArs?`

Returns

Current horizontal bar cursor settings.

Examples

`CURSor:HBArs?` might return the horizontal bar setting as return the horizontal bar setting as `CURSOR:HBARS:DELTA 0.0E+0;POSITION1 320.0000E+0;POSITION2 320.0000E+0;UNITS BASE`.

CURSor:HBArs:DELTa?

Returns the difference (in vertical units) between the two horizontal bar cursors in the instrument display. Query only.

Group

Cursor

Syntax

`CURSor:HBArs:DELTa?`

Related commands

CURSor:HBArs:UNIts on page 50

Returns

`<NR3>` is the difference between the horizontal bar cursors.

Examples

`CURSOR:HBARS:DELTA?` might return `:CURSOR:HBARS:DELTA 5.0800E+00` indicating that the difference between the two cursors is 5.08.

CURSor:HBArs:POSITION<x>

Sets or returns the horizontal bar cursor position relative to ground, which is expressed in vertical units (usually volts). The cursor is specified by x, which can be 1 or 2.

Group

Cursor

Syntax

`CURSor:HBArs:POSITION<x> <NR3>`

`CURSor:HBArs:POSITION<x>?`

Related commands

CURSor:FUNCtion on page 48

Arguments

`<NR3>` specifies the horizontal bar cursor position, relative to ground (in volts when the units are volts and amps when the units are amps), relative to the center of the screen (in divs when units are divisions), or relative to 1 V RMS (in decibels when the source is an FFT math waveform), for the waveform specified by the CURSor:SELect:SOUrce command.

The cursor position is limited to the graticule whenever an attempt is made to move it outside the graticule.

Note: The source determines the measurement units.

Examples

CURSOR:HBARS:POSITION1 25.0E-3 positions Cursor 1 of the horizontal cursors at 25 mV.

CURSOR:HBARS:POSITION2? might return :CURSOR:HBARS:POSITION2 -64.0000E-03 indicating that Cursor 2 of the horizontal bar cursors is at -64 mV.

CURSor:HBArs:UNIts

Sets or queries the vertical scale units for the selected cursor source waveform.

Group

Cursor

Syntax

CURSor:HBArs:UNIts {BASe|PERcent}

CURSor:HBArs:UNIts?

Arguments

BASe selects the vertical units for the selected waveform.

PERcent selects ratio cursors.

Examples

CURSor:HBArs:UNIts? might return :CURSOR:HBARS:UNITS BASE indicating that the units for the horizontal bar cursors are base.

CURSor:HBArs:USE

Sets the horizontal bar cursor measurement scale. This command is only applicable when ratio cursors are on. No query form.

Group

Cursor

Syntax

CURSor:HBArs:USE {CURrent | HALFgrat | FIVEdivs}

Related commands

CURSor:HBArs:UNIts on page 50

Arguments

CURrent sets the H Bar measurement scale so that 0% is the current position of the lowest H Bar cursor and 100% is the current position of the highest H Bar cursor.

HALFgrat resets the H bar measurement scale to half the number of divisions (five for some models and four for others) so that 25% is the current position of the lowest H Bar cursor and 75% is the current position of the highest H Bar.

FIVEdivs sets H Bar measurement scale so that five screen major divisions is 100%, where 0% is –2.5 divisions and 100% is +2.5 divisions from the center horizontal graticule.

Examples

CURSOR:HBARS:USE FIVEDIVS sets the H Bar measurement scale so that 5 screen major divisions equals 100%.

CURSor:MODe

Sets or returns whether the two cursors move linked together in unison or separately. This applies to the Waveform cursors display mode.

Conditions

This command is only applicable when waveform cursors are displayed.

Group

Cursor

Syntax

CURSor:MODe {TRACk|INDependent}

CURSor:MODe?

Arguments

TRACk ties the navigational functionality of the two cursors together. For cursor 1 adjustments, this ties the movement of the two cursors together; however, cursor 2 continues to move independently of cursor 1.

INDependent allows independent adjustment of the two cursors.

Examples

CURSOR:MODE TRACK specifies that the cursor positions move in unison.

CURSOR:MODE? might return :CURSOR:MODE TRACK indicating that the two cursors move in unison.

CURSor:VBArs?

Returns the current vertical bar cursor horizontal position and units settings. Query only.

Group

Cursor

Syntax

CURSor:VBArs?

Examples

CURSor:VBArs? might return CURSOR:VBARS:POSITION1 -7.680E-6;POSITION2 7.680E-6;DELTA 15.3600E-6;HPOS1 0.0E+0;HPOS2 0.0E+0;UNITS SECONDS;VDELTA 0.0E+0.

CURSor:VBArs:ALTERNATE<x>?

Returns the alternate readout for the waveform (Vbar) cursors specified by <x>. This alternate readout is in effect for a bus waveform. Query only.

Group

Cursor

Syntax

`CURSor:VBArs:ALTERNATE<x>?`

Arguments

`X = 1` specifies vertical bar cursor 1.

`X = 2` specifies vertical bar cursor 2.

Examples

`CURSor:VBArs:ALTERNATE1?` might return 1.001 indicating the vertical bar cursor 1 readout is 1.001.

CURSor:VBArs:DELTa?

Returns the time or frequency difference between the two vertical bar cursors. The units (seconds or Hertz) are specified by the CURSor:VBArs:UNIts command. If the cursor source is an FFT math waveform, CURSor:VBArs:DELTa is always in Hertz, regardless of the value set by CURSor:VBArs:UNIts. Query only.

Note: If Trigger View is active, this query returns 9.9E37 and generates event 221 (Settings conflict).

Group

Cursor

Syntax

`CURSor:VBArs:DELTa?`

Returns

`<NR3>`

Examples

`CURSor:VBArs:DELTa?` might return `8.92E-1`, indicating that the time difference between the vertical bar cursors is 0.892 seconds.

CURSor:VBArs:HPOS<x>?

Returns the horizontal value of the specified vertical bar ticks for cursor <x>. The units are specified by the CURSor:HBArs:UNIts query. <x> specifies the cursor. Valid values are 1 and 2. Query only.

Group

Cursor

Syntax

`CURSor:VBArs:HPOS<x>?`

Related Commands

CURSor:HBArs:UNIts on page 50

Returns

<x> indicates the cursor. Valid values are 1 and 2.

Examples

`CURSOR:VBARS:HPOS1?` might return `CURSOR:VBARS:HPOS2 100E-3`, indicating the value of one vertical bar tick.

CURSor:VBArs:POSITION<x>

Positions a vertical bar cursor. The unit is specified by the CURSor:VBArs:UNIts command, and can be in units of seconds or frequency (Hertz). If the cursor source is an FFT math waveform, CURSor:VBArs:POSITION is always in Hertz, regardless of the value set by CURSor:VBArs:UNIts.

Note: If Trigger View is active, the query form returns 9.9E37 and generates event 221 (Settings conflict).

Group

Cursor

Syntax

`CURSor:VBArs:POSITION<x>`

`CURSor:VBArs:POSITION<x>?`

Arguments

<x> specifies which cursor to position. Correct values are 1 and 2.

<NR3> specifies the cursor position in the units specified by the CURSor:VBArs:UNIts command. The position is relative to the trigger except when the cursor source is a math FFT waveform. The cursor position is limited to the graticule whenever an attempt is made to move it outside the graticule.

Examples

`CURSOR:VBARS:POSITION2 9.00E-6` positions the second vertical bar cursor at 9ms.

CURSOR:VBARS:POSITION1? might return 1.00E-6, indicating the first vertical bar cursor is at 1 µs.

CURSor:VBArs:UNIts

Sets or queries the units for the vertical bar cursors.

Note: When Trigger View is active, CURSor:VBArs:UNIts? generates event 221(Settings conflict).

Group

Cursor

Syntax

CURSor:VBArs:UNIts

CURSor:VBArs:UNIts?

Arguments

SECOnds specifies units of time.

HERtz specifies units of frequency (reciprocal of time).

Examples

CURSor:VBArs:UNIts SECONDS sets the units for the vertical bar cursors to seconds.

CURSor:VBArs:UNIts? returns HERTZ when the vertical bar cursor units are Hertz.

CURSor:VBArs:VDELTa?

Returns the vertical (amplitude) difference between the two vertical bar cursors. The units are specified by the CURSor:HBArs:UNits query. Query only.

Group

Cursor

Syntax

CURSor:VBArs:VDELTa?

Related commands

CURSor:HBArs:UNIts on page 50

Returns

<NR3> indicates the vertical difference between the two vertical bar cursors.

Examples

CURSor:VBArs:VDELTa? might return :CURSOR:VBARS:VDELTA 1.064E+0, indicating that the vertical difference between the vertical bar cursors is 1.064 units.

CURVe

Transfers instrument waveform data to and from the instrument in binary or ASCII format. Each waveform that is transferred has an associated waveform preamble that contains information such as data format, scale, and associated information.

For analog waveforms, the CURVe? query sends data from the instrument to an external device. The data source is specified by the DATa:SOUrce command. The first and last data points that are transferred are specified by the DATa:STARt and DATa:STOP commands.

Note: If the waveform specified by the DATa:SOUrce command is not displayed, the CURVe? query returns nothing, and generates events 2244 (Waveform requested is not activated) and 420 (Query UNTERMINATED).

The instrument returns data from the last acquisition if the source is a channel waveform that is being previewed. The data does not reflect the acquisition preview parameters. Your should always follow acquisition parameter changes with a single sequence OPC command prior to CURVe? to ensure the return data reflects the new acquisition parameters.

The CURVe command transfers waveform data from an external device to the instrument. The data is stored in the waveform location specified by DATa:DESTination, starting with the data point specified by DATa:STARt. Only one waveform can be transferred at a time. The waveform will only be displayed if the reference waveform is displayed.

Refer to *Waveform Commands* for a description of the waveform transfer process. *Waveform command group*

Group

Waveform

Syntax

CURVe {<Block>|<asc curve>}

CURVe?

Related Commands

DATa on page 57, *DATa:STARt* on page 58, *DATa:STOP* on page 59, *WFMInpre?* on page 164, *WFMInpre:BYT_Nr* on page 165, *WFMOutpre?* on page 170, *HEADer* on page 87

Arguments

<Block> is the waveform data in binary format. The waveform is formatted as:

#<x><yyy><data><newline>, where:

<x> is the number of y bytes. For example, if <yyy>=500, then <x>=3.

<yyy> is the number of bytes to transfer if samples are one or two bytes wide. Use the WFMInpre:BYT_Nr command to set the width for waveforms transferred into the instrument. Use WFMOutpre:BYT_Nr to set the width for waveforms transferred out from the instrument.

<data> is the curve data.

<newline> is a single byte new line character at the end of the data.

<asc curve> is the waveform data in ASCII format. The format for ASCII data is <NR1>[,<NR1>...] where each <NR1> represents a data point.

Examples

`CURVe?` with ASCII encoding, start and stop of 1 and 10 respectively, and a width set to 1 might return the following ASCII data:

`:CURVE 61,62,61,60,60,-59,-59,-58,-58,-59.`

D commands

This section lists commands and queries that begin with the letter D.

DATa

Sets or queries the format and location of the waveform data that is transferred with the CURVe command.

Group

Waveform

Syntax

`DATa {INIT|SNAp}`

`DATa?`

Related Commands

CURVe on page 55, *DATa:STARt* on page 58, *DATa:STOP* on page 59, *WFMInpre:NR_Pt?* on page 166, *WFMOutpre:NR_Pt?* on page 173

Arguments

`INIT` reinitializes the waveform data settings to their factory defaults except for DATa:STOP, which is set to the current acquisition record length.

`SNAp` sets `DATa:STARt` and `DATa:STOP` to match the current waveform cursor positions.

Examples

`DATa INIT` initializes the waveform data settings to their factory defaults.

`DATa?` might return `:DATA:DESTINATION REF1:ENCDG RIBINARY;SOURCE CH1;START 1;STOP 500;WIDTH 1`.

DATa:DESTination

Sets or queries the reference memory location for storing waveform data that is transferred into the instrument by the CURVe command.

Group

Waveform

Syntax

`DATa:DESTination REF<x>`

`DATa:DESTination?`

Related Commands

CURVe on page 55

Arguments

REF<x> is the reference memory location where the waveform will be stored.

Examples

DATa:DESTination REF1 stores incoming waveform data into reference memory 1.

DATa:DESTination? might return :DATA:DESTINATION REF2 indicating that reference 2 is the currently selected reference memory location for incoming waveform data.

DATa:SOUrce

Sets or queries which waveform will be transferred from the instrument by the CURVe? query. You can transfer only one waveform at a time.

Group

Waveform

Syntax

DATa:SOUrce{CH1|CH2| MATH|REF1|REF2}

DATa:SOUrce?

Related Commands

CURVe on page 55

Arguments

CH1– CH2 specifies which analog channel data will be transferred from the instrument to the controller, channels 1 through 2.

MATH specifies that the math waveform data will be transferred from the instrument to the controller.

REF1–REF2 specifies which reference waveform data will be transferred from the instrument to the controller, waveforms, 1 or 2.

Examples

DATA:SOURCE CH1 specifies that the channel 1 waveform will be transferred in the next CURVe? query.

DATa:SOUrce REF1 specifies that reference waveform REF1 will be transferred in the next CURVe? query.

DATa:SOUrce? might return :DATA:SOURCE REF2 indicating that the source for the waveform data which is transferred using a CURVe? query is reference 2.

DATa:STARt

Sets or queries the starting data point for incoming or outgoing waveform transfer. This command lets you transfer partial waveforms to and from the instrument.

Group

Waveform

Syntax

DATa:STARt <NR1>

DATa:STARt?

Related Commands

CURVe on page 55, *DATa* on page 57, *DATa:STOP* on page 59, *WFMInpre:NR_Pt?* on page 166, *WFMOutpre:NR_Pt?* on page 173

Arguments

<NR1> is the first data point that will be transferred, which ranges from 1 to the record length. Data will be transferred from <NR1> to DATa:STOP or the record length, whichever is less. If <NR1> is greater than the record length, the last data point in the record is transferred. DATa:STARt and DATa:STOP are order independent. When DATa:STOP is greater than DATa:STARt, the values will be swapped internally for the CURVE? query.

Examples

DATa:STARt 10 specifies that the waveform transfer will begin with data point 10.

DATa:STARt? might return :DATA:START 214 indicating that data point 214 is the first waveform data point that will be transferred.

DATa:STOP

Sets or queries the last data point in the waveform that will be transferred when using the CURVe? query. This lets you transfer partial waveforms from the instrument

Changes to the record length value are not automatically reflected in the DATa:STOP value. As record length is varied, the DATa:STOP value must be explicitly changed to ensure the entire record is transmitted. In other words, curve results will not automatically and correctly reflect increases in record length if the distance from DATa:STARt to DATa:STOP stays smaller than the increased record length.

When using the CURVe command, the instrument stops reading data when there is no more data to read.

Group

Waveform

Syntax

DATa:STOP <NR1>

DATa:STOP?

Related Commands

CURVe on page 55, *DATa* on page 57, *DATa:STOP* on page 59, *WFMInpre:NR_Pt?* on page 166, *WFMOutpre:NR_Pt?* on page 173

Arguments

<NR1> is the last data point that will be transferred, which ranges from 1 to the record length. If <NR1> is greater than the record length, then data will be transferred up to the record length. If both DATa:STARt and DATa:STOP are greater than the record length, the last data point in the record is returned.

DATa:STARt and DATa:STOP are order independent. When DATa:STOP is less than DATa:STARt, the values will be swapped internally for the CURVE? query.

If you always want to transfer complete waveforms, set DATa:STARt to 1 and DATa:STOP to the maximum record length, or larger.

Examples

DATa:STOP 15000 specifies that the waveform transfer will stop at data point 15000.

DATa:STOP? might return :DATA:STOP 14900 indicating that 14900 is the last waveform data point that will be transferred.

DATa:WIDth

Sets or queries the number of bytes per data point in the waveform transferred using the CURVe command.

Changes to the record length value are not automatically reflected in the DATa:STOP value. As record length is varied, the DATa:STOP value must be explicitly changed to ensure the entire record is transmitted. In other words, curve results will not automatically and correctly reflect increases in record length if the distance from DATa:STARt to DATa:STOP stays smaller than the increased record length.

Group

Waveform

Syntax

DATa:WIDth <NR1>

DATa:WIDth?

Related Commands

CURVe on page 55

Arguments

<NR1> = 1 sets the number of bytes per waveform data point to 1 byte (8 bits).

<NR1> = 2 sets the number of bytes per waveform data point to 2 bytes (16 bits). If DATa:WIDth is set to 2, the least significant byte is always zero. This format is useful for AVErage waveforms.

Examples

DATa:WIDth 1 sets the data width to 1 byte per data point for CURVe data.

DATE

Sets or queries the instrument date value. The instrument uses these values to time stamp files saved to the USB flash drive, as well as show the time and date on the instrument display.

Group

Miscellaneous

Syntax

DATE

DATE?

Related Commands

TIMe on page 145

Arguments

<QString> is a date in the form "yyyy-mm-dd".

Examples

DATE "2010-05-06" sets the date to May 6th, 2010.

DATE? might return :DATE 2015-10-29 indicating that the current date is set to Oct. 29, 2015.

DESE

Sets or queries the bits in the Device Event Status Enable Register (DESER). The DESER is the mask that determines whether events are reported to the Standard Event Status Register (SESR), and entered into the Event Queue. For a detailed discussion of the use of these registers, see Registers.

Group

Status and Error

Syntax

DESE <NR1>

DESE?

Related Commands

**CLS* on page 46, **ESE* on page 69, **ESR?* on page 70, *EVENT?* on page 70, *EVMsg?* on page 71, **SRE* on page 143, **STB?* on page 144

Arguments

<NR1> is an integer value in the range from 0 to 255. The binary bits of DESER are set according to this value. For example, DESE 209 sets the DESER to the binary value 11010001 (that is, the most significant bit in the register is set to 1, the next most significant bit to 1, the next bit to 0, and so on).

The power-on default for DESER is all bits set to 1 if *PSC is 1. If *PSC is 0, the DESER maintains its value through a power cycle.

Note: Setting DESER and ESER to the same value allows only those codes to be entered into the Event Queue and summarized on the ESB bit (bit 5) of the Status Byte Register. Use the *ESE command to set ESER. For more information on event handling, refer to the Status and Events section.

Examples

DESE 209 sets the DESER to binary 11010001, which enables the PON, URQ, EXE, and OPC bits.

DESE? might return the following string :DESE 186 , showing that DESER contains the binary value 10111010.

DIAg:FAN

Returns the currently set PWM fan value. Only query.

Group

Calibration and diagnostic

Syntax

DIAg:FAN?

Examples

DIAG:FAN? might return PWM=10, VOL=7.4V.

DIAg:LOOP:OPTion

Sets the self-test loop option.

Group

Calibration and diagnostic

Syntax

DIAg:LOOP:OPTion {ALWAYS|FAIL|ONFAIL|ONCE|NTIMES}

Arguments

ALWAYS continues looping until the self tests (diagnostics) are stopped via the front panel or by an instrument command.

FAIL causes looping until the first self test (diagnostic) failure or until self tests (diagnostics) are stopped.

ONFAIL causes looping on a specific test group as long as a FAIL status is returned from the test.

ONCE executes self test (diagnostics test) sequence once.

NTIMES runs “n” number of loops.

Examples

DIAg:LOOP:OPTion ONCE sets diagnostics to run one loop of self tests.

DIAg:LOOP:OPTion:NTIMes

Sets the self-test loop option to run N times.

Group

Calibration and diagnostic

Syntax

DIAg:LOOP:OPTion:NTIMes <NR1>

DIAg:LOOP:OPTion:NTIMes?

Arguments

<NR1> is the number of self-test loops.

Examples

DIAG:LOOP:OPTION:NTIMES 3 sets the self-test loop to run three times.

DIAG:LOOP:OPTION:NTIMES? might return :DIAG:LOOP:OPTION:NTIMES 5, indicating that the self-test loop is set to run five times.

DIAg:LOOP:STOP

Stops the self-test at the end of the current loop. No query form.

Group

Calibration and diagnostic

Syntax

DIAg:LOOP:STOP

Examples

DIAG:LOOP:STOP stops the self test at the end of the current loop.

DIAg:RESUlt:FLAg?

Returns the Pass/Fail status from the last diagnostic test sequence execution (those run automatically at power on, or those requested through the Service Menu). Use the DIAg:RESUlt:LOG? query to determine which test(s) has failed. Query only.

Group

Calibration and Diagnostic

Syntax

DIAg:RESUlt:FLAg?

Returns

PASS means that the instrument passes all selected diagnostic tests.

FAIL means that the instrument has failed at least one of the diagnostic tests.

Examples

DIAg:RESUlt:FLAG

Returns either PASS or FAIL.

DIAg:RESUlt:LOG?

Returns the internal results log from the last diagnostic test sequence execution (those run automatically at power on, or those requested through the Service Menu). The list contains all modules and module interfaces that were tested with the pass or fail status of each. Query only.

Group

Calibration and Diagnostic

Syntax

`DIAg:RESUlt:LOG?`

Returns

`<QString>` in the following format:

<Status>,<Module name>[,<Status>,<Module name>...]

Examples

`DIAg:RESUlt:LOG?` might return `:DIAG:RESULT:LOG "NOT RUN--CPU,NOT RUN--DISPLAY,NOT RUN--FPANEL,NOT RUN--IO,NOT RUN--ACQ,NOT RUN--RO M,NOT RUN--APPKEY"` for power-up diagnostics.

DIAg:SELect

Sets the type of diagnostics grouping. No query form.

Group

Calibration and diagnostic

Syntax

`DIAg:SELect {ALL|APPKey|CPU|DISplay|FPAnel|IO|ROM|ACQ}`

Arguments

`ALL` runs all diagnostic groups.

`CPU` runs just the CPU diagnostic group.

`DISplay` runs just the display circuit diagnostic group.

`FPAnel` runs just the front panel diagnostic group.

`IO` runs just the IO board diagnostic group.

`ROM` runs just the IO board diagnostic group.

`ACQ` runs just the acquisition system diagnostic group.

Examples

`DIAg:SELect ALL` runs all diagnostic groups.

DIAg:SELect:<function>

Runs self-tests on the specified system subsystem. No query form.

Group

Calibration and diagnostic

Syntax

`DIAg:SELect:<function>`

Arguments

`<function>` specifies a single instrument subsystem on which to run self tests (diagnostics). Valid values are:

- ACQ tests the acquisition system.
- CPU tests the CPU.
- DISplay tests the display.
- FPAnel tests the front panel controls.
- IO tests the IO ports.
- ROM tests the system read only memory.

Examples

`DIAg:SELect:ACQ` specifies to run self tests on the acquisition system.

DIAg:STATE

This command starts or stops the instrument self-test. Depending on the argument, self-test capabilities are either turned on or off. No query form.

Group

Calibration and diagnostic

Syntax

DIAg:STATE {EXECute|ABORt}

Arguments

`EXECute` starts diagnostics.

`ABORt` stops diagnostics at the end of the current loop.

Examples

`DIAG:STATE EXECute` starts diagnostics.

DIAg:TEMPVAL

Read out the currently FPGA chip and ambient temperature. Only query.

Group

Calibration and diagnostic

Syntax

DIAg:TEMPVAL?

Examples

`DIAG:TEMPVAL?` might return: `VDC Temp=-256, Ambient Temp=32`

DISplay:GRAticule

Sets and returns the display graticule intensity settings.

Group

Miscellaneous

Syntax

`DISplay:GRAticule {<NR1>|ON|OFF}`

`DISplay:GRAticule?`

Arguments

`ON` or `<NR1> ≠ 0` turns on the graticule in the screen display.

`OFF` or `<NR1> = 0` turns off the graticule in the screen display.

Examples

`DISPLAY:GRATICULE 0` sets `NO` graticule to display.

`DISPLAY:GRATICULE?` might return `:DISPLAY:GRATICULE 1` indicating that the graticule is on.

DISplay:INTENSITy:BACKLight

Sets and returns the waveform backlight intensity settings.

Group

Miscellaneous

Syntax

`DISplay:INTENSITy:BACKLight <NR1>`

`DISplay:INTENSITy:BACKLight ? <NR1>`

Arguments

<NR1> specifies the range from 10 to 100 .

Examples

DISplay:INTENSITy:BACKLight <NR1>

DISPLAY:INTENSITY:BACKLIGHT? might return :DISPLAY:INTEnsITY:BACKLIGHT 60

DISplay:PERSistence:STATe

Sets or returns the display persistence to ON or OFF. The query form returns the persistence state. This affects the display only.

Group

Display

Syntax

DISplay:PERSistence:STATe {ON | OFF | <NR1>}

DISplay:PERSistence:STATe?

Arguments

ON turns on the persistence on for the display.

OFF turns the persistence off for the display.

<NR1>=0 turns off the persistence for the display; any other value turns on the persistence for the display.

Examples

DISplay:PERSistence:STATe ON turns on the persistence on the display for all waveforms.

DISplay:PERSistence:STATe? might return 1 indicating that the persistence is ON for the display.

DISplay:PERSistence:VALUe

Sets the value of persistence if the persistence state is set to on. This affects the display only.

Group

Display

Syntax

DISplay:PERSistence:VALUe {<NR3> | AUTO | INFInite}

DISplay:PERSistence:VALUe?

Arguments

<NR3> specifies the time of the persistence.

INFInite displays waveform points until a control change resets the acquisition system. When persistence is set to infinite, it does not mean that the brightness of any pixel should never decrease. The brightness of a pixel is proportionally dependent on the ratio between the intensity, which does NOT decrease at infinite persistence, and the maximum value of intensity of any pixel on the screen. Thus, if a particular pixel gets hit less often than others, its brightness will decrease over time. It will become less bright relative to the pixels that get hit often. AUTO specifies that the oscilloscope automatically determines the best waveform persistence based on the value of waveform intensity (DISPLAY:INTENSITY:WAVEFORM)

Examples

`DISplay:PERSistence:VALUe 3` specifies that the waveform points are displayed fading for 3 seconds before they completely disappear.

E commands

This section lists commands and queries that begin with the letter E.

ERRLOG:FIRST?

Returns the first entry in the error log, or an empty string if the error log is empty. Use this command with ERRLOG:NEXT? to retrieve error log messages. Query only.

Group

Calibration and Diagnostic

Syntax

```
ERRLOG:FIRST?
```

Returns

Refer to the service manual for your instrument for information about error log message format.

ERRLOG:NEXT?

Returns the next entry in the error log, or an empty string if the error log is empty or you have reached the end of the log. To start at the top of the error log, run the ERRLOG:FIRST? query to return the first error log message. Then use the ERRLOG:NEXT? query to step through the error log. Query only.

Group

Calibration and Diagnostic

Syntax

```
ERRLOG:NEXT?
```

Returns

Refer to the service manual for your instrument for information about error log message format.

*ESE

Sets and queries the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (STB). For a detailed discussion on how to use registers, see Registers. Command only, no query form.

Group

Status and Error

Syntax

```
*ESE <NR1>
```

*ESE?

Related Commands

*CLS on page 46, DESE on page 61, *ESR? on page 70, EVENT? on page 70, EVMsg? on page 71, *SRE on page 143, *STB? on page 144

Arguments

<NR1> is a value in the range from 0 through 255. The binary bits of the ESER are set according to this value.

The power-on default for ESER is 0 if *PSC is 1. If *PSC is 0, the ESER maintains its value through a power cycle.

Note: Setting the DESER and the ESER to the same value allows only those codes to be entered into the Event Queue and summarized on the ESB bit (bit 5) of the Status Byte Register. Use the DESE command to set the DESER. See*Event Handling Sequence* on page 186.

Examples

*ESE 209 sets the ESER to binary 11010001, which enables the PON, URQ, EXE, and OPC bits.

*ESE? might return the string *ESE 186, showing that the ESER contains the binary value 10111010.

*ESR?

Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). For a detailed discussion on how to use registers, see Registers. Query only.

Group

Status and Error

Syntax

*ESR?

Related Commands

ALLEv? on page 29, *CLS on page 46, DESE on page 61, *ESE on page 69, EVENT? on page 70, EVMsg? on page 71, *OPC on page 128, *SRE on page 143, *STB? on page 144

Returns

Contents of the Standard Event Status Register.

Examples

*ESR? might return the value 213, showing that the SESR contains binary 11010101.

EVENT?

Returns from the Event Queue an event code that provides information about the results of the last *ESR? read. EVENT? also removes the returned value from the Event Queue. Query only.

Group

Status and Error

Syntax

EVENT?

Related Commands

ALLEv? on page 29, **CLS* on page 46, *DESE* on page 61, **ESE* on page 69, **ESR?* on page 70, *EVMsg?* on page 71, **SRE* on page 143, **STB?* on page 144

Returns

<NR1> the last *ESR.

Examples

EVENT? might return EVENT 110, indicating there was an error in a command header.

EVMsg?

Removes from the Event Queue a single event code associated with the results of the last *ESR? read, and returns the event code with an explanatory message. Query only.

Group

Status and Error

Syntax

EVMsg?

Related Commands

ALLEv? on page 29, **CLS* on page 46, *DESE* on page 61, **ESE* on page 69, **ESR?* on page 70, *EVENT?* on page 70, **SRE* on page 143, **STB?* on page 144

Returns

The event code and message in the following format:

<Event Code><Comma><QString>[<Event Code><Comma> <QString>...]

<QString>::= <Message>;[<Command>]

where <Command> is the command that caused the error and may be returned when a command error is detected by the instrument. As much of the command as possible is returned without exceeding the 60 character limit of the <Message> and <Command> strings combined. The command string is right-justified.

Examples

EVMsg? might return the message EVMSG 110, "Command header error"

EVQty?

Returns the number of event codes that are in the Event Queue. This is useful when using ALLEv? since it lets you know exactly how many events will be returned. Query only.

Group

Status and Error

Syntax

`EVQty?`

Related Commands

ALLEv? on page 29, *EVENT?* on page 70, *EVMsg?* on page 71

Returns

`<NR1>` is the number of event codes in the Event Queue.

Examples

`EVQty?` might return `:EVQTY 3` indicating the number of event codes in the Event Queue is 3.

F commands

This section lists commands and queries that begin with the letter F.

FACtory

Resets the instrument to its factory default settings. Refer to Appendix B: Factory Setup for a list of the factory default settings. No query.

This command does the following:

- Clears the Event Status Enable Register
- Clears the Service Request Enable Register
- Sets the Device Event Status Enable Register to 255
- Purges all defined aliases
- Enables all Command Headers
- Sets the macro defined by *DDT to a "zero-length field
- Clears the pending operation flag and associated operations

This command does not reset the following:

- Communication settings
- Sate of the VXI-11 (Ethernet IEEE Std 488.2) interface
- Calibration data that affects device specifications
- Protected user data
- Stored settings
- Power On Status Clear Flag
- instrument password

Group

Save and Recall

Syntax

`FACtory`

Related Commands

*PSC on page 129, *RCL on page 130, RECAll:SETUp on page 130, *RST on page 134, *SAV on page 135, SAVe:SETUp on page 137, SAVe:IMAge:FILEFormat on page 136

Examples

`FACTORY` resets the instrument to its factory default settings. Refer to *Factory Setup*.

FEAEN:PASSWORD

Checks if the password entered is equal to the preset password for Feature Enable which allows the educators to enable or disable features such as Autoset, Cursors or Measurements. The same can be set manually from the Utility menu when we try to change the settings of Autoset Enable, Cursors Enable or Measurement Enable from Feature Enable. No query form. To access the menu, refer product user manual.

Group

Miscellaneous

Syntax

```
FEAEN:PASSWORD <password>
```

Related Commands

AUTOSet:ENABLE on page 30

CURSor:ENABLE on page 47

MEASUrement:ENABLE

Arguments

<password> is the feature enable password, enclosed in quotes.

Examples

`FEAEN:PASSWORD "111111"` allows the user to change the settings of Feature Enable if the feature enable password matches the string "111111".

FFT?

Returns all FFT parameters. Query only.

Group

FFT

Syntax

```
FFT?
```

Related commands

FFT:VERTical:SCAle on page 77, *FFT:VERTical:POSition* on page 76, *FFT:VERTical:UNIts* on page 77, *FFT:HORizontal:SCAle* on page 75, *FFT:HORizontal:POSition* on page 74 , *FFT:SOURce* on page 75, *FFT:SRCWFM* on page 76, *FFT:WINdow* on page 78, *SELect:FFT* on page 140

Examples

`FFT?` might return `1; CH1; 20; 0.000; "dB"; 250.000E+3; 750.000E+3; "Hz";1;HANNING;X1;DB`

FFT:HORizontal:POSition

Sets or queries the FFT horizontal display position.

Group

FFT

Syntax

`FFT:HORizontal:POSition <NR3>`

`FFT:HORizontal:POSition?`

Arguments

`<NR3>` is the FFT horizontal display position.

Examples

`FFT:HORizontal:POSition 750.0E+3` sets the FFT horizontal position to 750.0E+3.

`FFT:HORizontal:POSition?` might return `750.000E+3`.

FFT:HORizontal:SCAle

Sets or queries the horizontal scale of the FFT waveform.

Group

FFT

Syntax

`FFT:HORizontal:SCAle <NR3>`

`FFT:HORizontal:SCAle?`

Arguments

`<NR3>` is the FFT horizontal scale.

Examples

`FFT:HORizontal:SCALe 500.00E+6` sets the FFT horizontal scale to 500 MHz.

`FFT:HORizontal:SCALe?` might return `500.00E+6` indicating the FFT horizontal scale is set to 500 MHz.

FFT:SOURce

Sets or queries the source of the FFT waveform.

Group

FFT

Syntax

`FFT:SOURce {CH1|CH2}`

`FFT:SOURce?`

Arguments

{CH1|CH2} the FFT source channel.

Examples

FFT:SOURce ch2 sets the FFT source waveform to CH2.

FFT:SOURce? might return "CH2" if CH2 is the FFT source waveform.

FFT:SRCWFM

Sets or queries the FFT source waveform display state.

Group

FFT

Syntax

FFT:SRCWFM <ON|OFF|NR1>

FFT:SRCWFM?

Arguments

<NR1> = 0 does not display the FFT source waveform, any other value displays the FFT source waveform.

Examples

FFT:SRCWFM 0 turns off the display of the FFT source waveform.

FFT:SRCWFM? might return 1 indicating the FFT source waveform is displayled.

FFT:VERTical:POSition

Sets or queries the FFT vertical display position.

Group

FFT

Syntax

FFT:VERTical:POSition <NR2>

FFT:VERTical:POSition?

Arguments

<NR2> is the FFT vertical position.

Examples

FFT:VERTical:POSition 2 sets the FFT vertical position to 2 divisions above center screen.

FFT:VERTical:POSition? might return 2.000.

FFT:VERTical:SCAle

Sets or queries the FFT vertical zoom factor.

Group

FFT

Syntax

FFT:VERTical:SCAle <NR2>

FFT:VERTical:SCAle?

Arguments

<NR2> is the FFT vertical scale.

Examples

FFT:VERTical:SCAle 20 sets the FFT waveform vertical scale to 20.

FFT:VERTical:SCAle? might return 20.00 indicating the FFT waveform vertical scale is 20 dB.

FFT:VERTical:UNIts

Queries the FFT vertical measurement units label.

Group

FFT

Syntax

FFT:VERTical:UNIts?

Examples

FFT:VERTical:UNIts? might return dB indicating the FFT vertical units are set to dB.

FFT:VType

Sets or queries the FFT waveform vertical units.

Group

FFT

Syntax

FFT:VType {DB|LINEAr}

FFT:VType?

Examples

FFT:VType DB sets the FFT waveform vertical units to dB.

FFT:VType? might return DB.

FFT:WINdow

Sets or queries the FFT window type.

Group

FFT

Syntax

FFT:WINdow {HAMming|HANning|RECTangular|BLAckmanharris}

FFT:WINdow?

Arguments

RECTangular window function is equivalent to multiplying all gate data by one.

HAMming window function is based on a cosine series.

HANning window function is based on a cosine series.

BLAckmanharris window function is based on a cosine series.

Examples

FFT:WINdow HAMMING sets the FFT window to Hamming.

FFT:WINdow? might return HAMMING.

FILESystem?

Returns the current working directory and amount of free space. This query is the same as the FILESystem:DIR? query and the FILESystem:FREESpace? query. Query only.

.

Group

File system

Syntax

FILESystem

FILESystem?

Related commands

FILESystem:CWD on page 79, *FILESystem:DELEte* on page 79, *FILESystem:DIR?* on page 80, *FILESystem:REName* on page 82

Examples

`FILESYSTEM?` might return `:FILESYSTEM:DIR "TEK00000.BMP","GLITCH1.PNG","TEMP.TMP", "FILE1.WFM","FILE2.WFM", "MATH1.WFM"," REF1.WFM","REF2.WFM"`.

FILESystem:CWD

Sets or queries the current working directory (CWD) for FILESystem commands.

The default working directory is USB0. Anytime you use this command to change the directory, the directory that you specify is retained as the current working directory until you either change the directory or you delete the directory. If you delete the current working directory, the instrument resets current working directory to the default directory (USB0) the next time the instrument is powered on or the next time you execute a file system command.

This command supports the permutations of file and directory names supported by Microsoft Windows:

Relative path names; for example, "./temp"

Absolute path names; for example, " USB0/Wfms"

Implied relative path names; for example "NEWFILE.TXT" becomes " USB0/TEKSCOPE/NEWFILE.TXT" if the current working directory is " USB0/TEKSCOPE"

Group

File system

Syntax

`FILESystem:CWD {<new working directory path>}`

`FILESystem:CWD?`

Arguments

`<new working directory path>` is a quoted string that defines the current working; a directory name can have up to 8 characters with an extension of up to 3 characters.

Examples

`FILESYSTEM:CWD " USB0/TEKSCOPE/IMAGES"` sets the current working directory to images.

`FILESYSTEM:CWD?` might return `:FILESYSTEM:CWD " USB0/TEKSCOPE/WAVEFORMS"` indicating that the current working directory is set to waveforms.

FILESystem:DELEte

This command deletes a named file. If you specify a directory name, it will delete the directory and all of its contents, the same as the RMDir command. You can also specify the filename as *.* to delete all of the files in the current or specified directory. Command only, no query form.

Group

File system

Syntax

`FILESystem:DELEte <file path>`

Related commands

FILESystem:CWD on page 79, *FILESystem:RMDir* on page 83

Arguments

`<file path>` is a quoted string that defines the folder path and file name of the file to delete. If the file path is within the current working directory, you need only specify the file name. The argument *.* will delete all files and subdirectories within the current working directory.

Examples

`FILESYSTEM:DELETE "NOT_MINE.SET"` deletes the file named `NOT_MINE.SET` from the current working directory.

FILESystem:DIR?

Returns a list of quoted strings. Each string contains the name of a file or directory in the current working directory. Query only.

Group

File system

Syntax

`FILESystem:DIR?`

Related commands

FILESystem:CWD on page 79, *FILESystem:RMDir* on page 83

Returns

`FILESystem:DIR?` returns a list of files and directories in the current working directory.

Examples

`FILESystem:DIR?` might return `:FILESYSTEM:DIR "TEK00000.PNG","CANSETUP.SET","WFM1.ISF","MYIMAGES"`.

FILESystem:FORMat

Formats a mass storage device. This command should be used with extreme caution as it causes all data on the specified mass storage device to be lost. Drive letters (such as USB0) are case sensitive and must be upper case. For all other FILESYSTEM commands, drives letters are not case sensitive. Example: FILES:FORMAT " USB0/" Formats the USB flash drive installed in the instrument's front panel USB port. Command only, no query form.

Group

File system

Syntax

`FILESystem:FORMat <drive>`

Arguments

`<drive>` is a quoted string that sets the drive to format.

Examples

`FILESystem:FORMat "/usb0/"` formats the USB flash drive installed in the instrument front panel USB port.

FILESystem:FREESpace?

Returns a numeric value, in bytes, of the memory space available on the current drive. Query only.

Group

File system

Syntax

`FILESystem:FREESpace?`

Related commands

FILESystem:CWD on page 79

Examples

`FILESystem:FREESpace?` might return `6242501`.

FILESystem:MKDir

Creates a folder at the specified location. Command only, no query form.

Group

File system

Syntax

`FILESystem:MKDir <directory path>`

Related commands

FILESystem:CWD on page 79, *FILESystem:DIR?* on page 80

Arguments

<directory path> is a quoted string that defines the location and name of the directory to create. If you do not specify a path to the directory, the instrument creates the directory in the current working directory. The current directory refers to the name of a directory as returned by the FILESystem:CWD query.

Directory names must follow the same rules as file names. *File System Conventions*

Examples

FILESYSTEM:MKDIR " USB0/NewDirectory" creates the directory named NEWDIRECTORY at the root of the E drive.

The following two commands create the directory MYNEWSUBDIRECTORY within the existing directory mydirectory at the root of the USB0 drive: FILESYSTEM:CWD " USB0/MyDirectory";:FILESYSTEM:MKDIR "MyNewSubDirectory" This assumes that USB0/MYDIRECTORY already existed and was not a read-only directory.

FILESystem:READFile

Writes the contents of the specified file to the specified interface. If the file does not exist or is not readable, an appropriate error event is posted. No query form.

Group

File System

Syntax

FILESystem:READFile <QString>

Related commands

FILESystem:CWD on page 79

Arguments

<QString> is a quoted string that defines the file name and path. If the file path is within the current working directory, specify only the file name.

Examples

FILESYSTEM:READFILE "USB0/TEST_DATA/TEK00016CH1.CSV" reads the content of the specified file, if the file exists and is readable, and sends the content of the file to the current interface.

FILESystem:RENname

Assigns a new name to an existing file or folder. You can also move a file or folder by specifying the new name in a different folder. Command only, no query form.

For file and folder name rules, see *File System Conventions*. *File System Conventions*

Group

File system

Syntax

FILESystem:RENName <old file path>,<new file path>

Related commands

FILESystem:CWD on page 79

Arguments

<old filepath> is a quoted string that defines the path and name of the file to rename. If you do not specify a path to the file, the instrument looks for the file in the current working folder. The current directory refers to the name of a folder as returned by the FILESystem:CWD query.

<new filepath> is a quoted string that defines the path and new name of the file. If you do not specify a path to a folder, the instrument places the renamed file into the current working folder. *File System Conventions*

Examples

FILESYSTEM:RENAME " USB0/TEK00000.SET","D:/MYSETTING.SET" gives the file named TEK00000.SET the new name of MYSETTING.SET. The file remains in the root directory on the D drive.

FILESystem:RMDir

Deletes a named directory. This command deletes the specified directory and all of its contents. The directory must not be a read-only directory. Command only, no query form.

Group

File system

Syntax

FILESystem:RMDir <directory path>

Arguments

<directory path> is a quoted string that defines the location and name of the directory to delete. If you do not specify a path to the folder, the instrument deletes the specified folder in the current working folder. The current folder refers to the name of a folder as returned by the FILESystem:CWD query.

Note: A folder must be empty before you can delete it.

Examples

FILESYSTEM:RMDIR " USB0/OldDirectory" removes the directory named olddirectory from the root of the E drive.

FILESystem:WRITEFile

Writes the specified block data to a file in the instrument current working directory. If the specified file does not exist or is not readable, an appropriate error event is posted. The maximum length of the block data is 262144 bytes. No query form.

Group

File System

Syntax

```
FILESystem:WRITEFile <file path>, <data>
```

Related commands

FILESystem:CWD on page 79

Arguments

`<file path>` is the quoted string that defines the file name and path. If the path is within the current working directory, specify the file name only.

`<data>` can be either DEFINITE LENGTH encoding or INDEFINITE LENGTH ARBITRARY BLOCK PROGRAM DATA encoding as described in IEEE488.2.

FPAnel:PRESS

Simulates the action of pressing a specified front-panel button. No query form.

When the front panel is locked, the front-panel buttons and multipurpose knob operations are suspended. The `FPAnel:PRESS` and the `FPAnel:TURN` commands will also not work. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands.

Group

Miscellaneous

Syntax

```
FPAnel:PRESS <button>
```

Arguments

`<button>` is the name of a front-panel button. Most of the argument names associate directly with their front panel buttons. For example, `AUTOSet` is for the Autoset button. The <button> enumeration arguments and their associations with the front panel buttons are listed below.

Argument	Button
ACQuire	Acquire button
SAVE Recall	Save/Recall Menu button
MEASurement	Measure button
UTILity	Utility button

Table continued…

Argument	Button
MATh	M button
REF	R button
FFT	F button
TRIGger	Trigger Menu button
FORCetrig	Force Trig button
CH1	Channel1 select button
CH2	Channel2 select button
DEFaultsetup	Default Setup button
FUNCtion	Function button
Help	Help button
ZOOM	Zoom button
FINe	Fine button
CURsor	Cursors button
RUnstop	Run/Stop button
SINGleseq	Single button
AUTOset	Autoset button
SETTO50	Trigger level knob can be pressed to Set Trigger to 50%
Save	Save button
RMENU1	Screen top-most side menu button
RMENU2	Screen side menu button
RMENU3	Screen side menu button
RMENU4	Screen side menu button
RMENU5	Screen bottom-most side menu button
MENUOff	Menu On/Off button
GPKNOB	Multipurpose knob can be pressed for selection.
HORZPos	Horizontal Position knob can be pressed to set horizontal position to center.

Examples

`FPANEL:PRESS AUTOSET` executes the instrument Autoset function.

FPAnel:TURN

Simulates the action of turning a specified front-panel control knob. No query form.

When the front panel is locked, the front-panel button and multipurpose knob operations are suspended. The `FPAnel:PRESS` and `FPAnel:TURN` commands will also not work, and they will not generate an error. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands. For example, to set the trigger level to 50%, you could use `TRIGger:A SETLevel`. To force a trigger, you could use `TRIGger FORCe`.

Group

Miscellaneous

Syntax

`FPAnel:TURN <knob>,[<n>]`

Arguments

`<knob>` is the name of a rotating control. A comma (,) separates the control knob argument from the numeric optional rotation value argument. In the absence of the numeric rotation value argument, the default is 1 (clockwise). You do not need a white space between the arguments and the comma. <n> represents the rotation direction and magnitude of rotation. Negative values represent a counterclockwise knob rotation, and positive values represent a clockwise rotation. The magnitude of <n> specifies the amount of the turn, where <n> = 1 represents turning the knob one unit, <n> = 2 represents turning the knob two units, <n> = 4 represents turning the knob four units, and so on. The range of units depends on which front panel knob is specified.

Table 8: FPAnel:TURN arguments

Argument	Knob
GPKNOB	Multipurpose knob
HORZPos	Horizontal Position knob
HORZScale	Horizontal Scale knob
TRIGLevel	Trigger Level knob
VERTPOS<n>	Vertical Position knob
VERTSCALE<n>	Vertical Scale knob

Examples

`FPANEL:TURN TRIGLEVEL,10` duplicates turning the front panel Trigger Level knob clockwise by 10 units.

FWUpdate:Update

Updates the oscilloscope firmware from a file on a USB flash drive. Before executing this command, make sure the USB flash drive is plugged into the instrument, and contains the firmware update file **TBS1000C.TEK** at the root (top) directory. If the update file is not in the root directory, the oscilloscope shows a warning message and the firmware is not updated.

Group

Miscellaneous

Syntax

FWUpdate:Update

H commands

This section lists commands and queries that begin with the letter H.

HDR

This command is identical to the HEADer query and is included for compatibility with other Tektronix scopes.

Group

Miscellaneous

Syntax

`HDR`

`HDR?`

HEADer

Sets and queries the Response Header Enable State that causes the to either include or omit headers on query responses. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk); they never return headers.

Group

Miscellaneous

Syntax

`HEADer`

`HEADer?`

Arguments

`ON` or `<NR1>` ≠ 0 sets the Response Header Enable State to true. This causes the instrument to include header on applicable query responses. You can then use the query response as a command.

`OFF` or `<NR1>` = 0 sets the Response Header Enable State to false. This causes the instrument to omit headers on query responses so that only the argument is returned.

Examples

`HEADer OFF` causes the instrument to omit headers from query responses.

`HEADer?` might return `1`, showing that the Response Header Enable State is true. Query only.

HELPevery:ACQuire

Enables or disables the display of Help Everywhere information for the acquire measurements.

Group

Help everywhere

Syntax

HELPevery:ACQuire {ON|OFF}

HELPevery:ACQuire?

Arguments

`ON` enables displaying Help Everywhere for the acquire measurements.

`OFF` disables displaying Help Everywhere for the acquire measurements.

Examples

`HELPevery:ACQuire ON` enables Help Everywhere for the acquire settings.

HELPevery:ALL

Enables or disables the display of Help Everywhere information for all measurement settings (acquire, trigger, vertical, math, fft ,cursor, reference, measurement, and utility modules).

Group

Help everywhere

Syntax

`HELPevery:ALL {ON|OFF}`

Arguments

`ON` enables Help Everywhere.

`OFF` disables Help Everywhere.

Examples

`HELPevery:ALL ON` enables Help Everywhere.

HELPevery:CURsor

Enables or disables the display of Help Everywhere information for the cursor module.

Group

Help everywhere

Syntax

HELPevery:CURsor {ON|OFF}

HELPevery:CURsor?

Arguments

`ON` enables Help Everywhere for the cursor settings.

OFF disables Help Everywhere for the cursor settings.

Examples

HELPevery:CURsor ON enables Help Everywhere for the cursor settings.

HELPevery:FFT

Enables or disables the display of Help Everywhere information for the fft settings.

Group

Help everywhere

Syntax

HELPevery:FFT {ON|OFF}

HELPevery:FFT?

Arguments

ON enables Help Everywhere for the FFT module.

OFF disables Help Everywhere for the FFT module.

Examples

HELPevery:FFT ON enables Help Everywhere for the FFT module.

HELPevery:MATH

Enables or disables the display of Help Everywhere information for the math module.

Group

Help everywhere

Syntax

HELPevery:MATH {ON|OFF}

HELPevery:MATH?

Arguments

ON enables Help Everywhere for the math module.

OFF disables Help Everywhere for the math module..

Examples

HELPevery:MATH ON enables Help Everywhere for the math module.

HELPevery:MEASUrement

Enables or disables the display of Help Everywhere information for the measurement module.

Group

Help everywhere

Syntax

HELPevery:MEASUrement {ON|OFF}

HELPevery:MEASUrement?

Arguments

`ON` enables Help Everywhere for the measurement module.

`OFF` disables Help Everywhere for the measurement module..

Examples

`HELPevery:MEASUrement ON` enables Help Everywhere for the measurement module.

HELPevery:REFerence

Enables or disables the display of Help Everywhere information for the reference module.

Group

Help everywhere

Syntax

HELPevery:REFerence {ON|OFF}

HELPevery:REFerence?

Arguments

`ON` enables Help Everywhere for the reference module.

`OFF` disables Help Everywhere for the reference module.

Examples

`HELPevery:REFerence ON` enables Help Everywhere for the reference module.

HELPevery:TRIGger

Enables or disables the display of Help Everywhere information for the trigger module.

Group

Help everywhere

Syntax

HELPevery:TRIGger {ON | OFF}

HELPevery:TRIGger?

Arguments

ON enables Help Everywhere for the trigger module.

OFF disables Help Everywhere for the trigger module.

Examples

HELPevery:TRIGger ON enables Help Everywhere for the trigger module.

HELPevery:UTIlity

Enables or disables the display of Help Everywhere information for the utility module.

Group

Help everywhere

Syntax

HELPevery:UTIlity {ON|OFF}

HELPevery:UTIlity?

Arguments

ON enables Help Everywhere for the utility module.

OFF disables Help Everywhere for the utility module.

Examples

HELPevery:UTIlity ON enables Help Everywhere for the utility module.

HELPevery:VERtical

Enables or disables the display of Help Everywhere information for the vertical module.

Group

Help everywhere

Syntax

HELPevery:VERtical {ON|OFF}

HELPevery:VERtical?

Arguments

ON enables Help Everywhere for the vertical module.

OFF disables Help Everywhere for the vertical module.

Examples

HELPevery:VERtical ON enables Help Everywhere for the vertical module.

HORizontal?

Returns all settings for the horizontal commands. Query only.

The commands HORizontal:MAIn:SCAle, HORizontal:MAIn:SECdiv, HORizontal:SCAle, and HORizontal:SECdiv are equivalent, so HORizontal:MAIn:SCAle is the value that is returned. The commands HORizontal:MAIn:POSition and HORizontal:POSition are equivalent, so HORizontal:MAIn:POSition is the value that is returned.

Group

Horizontal

Syntax

HORizontal?

Returns

Returns all horizontal settings.

Examples

HORIZONTAL? might return the following horizontal settings :HORIZONTAL:POSITION 50.0000; SAMPLERATE 500.0000E+6; SCALE 200.0000E-9; RECORDLENGTH 2000; RECORDLENGTH : AUTO 0; DELAY:MODE 1; TIME 0.0E+0.

HORizontal:ACQLENGTH

Queries the record length. Query only.

Group

Horizontal

Syntax

HORizontal:ACQLENGTH?

Related commands

HORizontal:RECOrdlength on page 98

Examples

HORIZONTAL:ACQLENGTH? might return 20000 indicating that the record length is 20k points.

HORizontal:DIVisions

Returns the current horizontal divisions: 15.0000. Query only.

Group

Horizontal

Syntax

HORizontal:DIVisions?

Examples

HORizontal:DIVisions? might return 15.

HORizontal[:MAIn][:DELay]:POSition

Sets or queries the horizontal position. If Horizontal Delay Mode is turned off, this command is equivalent to adjusting the HORIZONTAL POSITION knob on the front panel. When Horizontal Delay Mode is on, this command stores a new horizontal position that is used when Horizontal Delay Mode is turned off.

Group

Horizontal

Syntax

HORizontal:MAIn:DELay:POSition <NR1>

HORizontal:MAIn:DELay:POSition?

HORizontal:MAIn:POSition <NR1>

HORizontal:MAIn:POSition?

HORizontal:DELay:POSition <NR1>

HORizontal:DELay:POSition?

HORizontal:POSition <NR1>

HORizontal:POSition?

Arguments

<NR1> is the horizontal position expressed as the percentage of the waveform displayed left of the center of the graticule.

Examples

HORIZONTAL:DELay:POSITION 50 sets the horizontal position to 50%.

HORIZONTAL:MAIn:DELay:POSITION? might return 100 indicating that the horizontal position is set to 100%.

HORizontal:MAIn:DELay:MODe

Sets or returns the horizontal delay mode.

Group

Horizontal

Syntax

`HORizontal:MAIn:DELay:MODe {OFF|ON|<NR1>}`

`HORizontal:MAIn:DELay:MODe?`

`HORizontal:DELay:MODe {OFF|ON|<NR1>}`

`HORizontal:DELay:MODe?`

Related Commands

HORizontal[:MAIn][:DELay]:POSition on page 93

Arguments

`OFF` sets the Horizontal Delay Mode to off. This causes the `HORizontal:POSition` command to horizontally position the waveform.

`ON` sets the Horizontal Delay Mode to on. This causes the `HORizontal:DELay:TIMe` command to horizontally position the waveform.

`<NR1> = 0` sets the Horizontal Delay Mode to off; any other value sets this mode to on.

Examples

`HORIZONTAL:DELAY:MODE OFF` sets the Horizontal Delay Mode to off, allowing the `HORizontal:POSition` command to horizontally position the waveform.

`HORIZONTAL:MAIn:DELAY:MODE?` might return `OFF` indicating that the `HORizontal:POSition` command horizontally positions the waveform.

HORizontal:MAIn:DELay:STATe

Sets or returns the horizontal delay state . The same as HORizontal[:MAIn]:DELay:MODe.

Group

Horizontal

Syntax

`HORizontal:MAIn:DELay:STATe {OFF|ON|<NR1>}`

`HORizontal:MAIn:DELay:STATe?`

`HORizontal:DELay:STATe {OFF|ON|<NR1>}`

`HORizontal:DELay:STATe?`

Related Commands

HORizontal[:MAIn][:DELay]:POSition on page 93

Arguments

OFF sets the Horizontal Delay State to off. This causes the HORizontal:POSition command to horizontally position the waveform.

ON sets the Horizontal Delay State to on. This causes the command to horizontally position the waveform.

<NR1> = 0 sets the Horizontal Delay State to off; any other value sets this mode to on.

Examples

HORIZONTAL:DELAY:STATE OFF sets the Horizontal Delay State to off, allowing the HORizontal:POSition command to horizontally position the waveform.

HORIZONTAL:MAIN:DELAY:STATE? might return OFF indicating that the HORizontal:POSition command horizontally positions the waveform.

HORizontal[:MAIn]:DELay:TIMe

Sets or queries the horizontal delay time. The amount of time the acquisition is delayed depends on sample rate and record length.

Group

Horizontal

Syntax

HORizontal:MAIn:DELay:TIMe <NR3>

HORizontal:MAIn:DELay:TIMe?

HORizontal:DELay:TIMe <NR3>

HORizontal:DELay:TIMe?

Arguments

<NR3> is the delay in seconds.

Examples

HORizontal:DELay:TIME 0.3 sets the delay of acquisition data so that the resulting waveform is centered 300 ms after the trigger occurs.

HORizontal[:MAIn]:SAMPLERate

Returns the current horizontal sample rate. Query only.

Group

Horizontal

Syntax

HORizontal:SAMPLERate?

HORizontal [:MAIn] :SAMPLERate?

Examples

HORizontal:SAMPLERate? might return 2.0000E+9.

HORizontal[:MAIn]:SCAle

Sets or queries the time base horizontal scale. Query only.

Group

Horizontal

Syntax

HORizontal:SCAle <NR3>

HORizontal:SCAle?

HORizontal:MAIn:SCAle <NR3>

HORizontal:MAIn:SCAle?

Arguments

<NR3> specifies the range from 2 ns to 100 s, depending on the model.

Returns

All settings for the time base

Examples

HORizontal:MAIn:SCAle 2E-6 sets the main scale to 2 µs per division.

HORizontal:MAIn:SCAle? might return 2.0000E-06, indicating that the main scale is currently set to 2 µs per division.

HORizontal[:MAIn]:SECdiv

Sets the time per division for the main time base. This command is identical to the HORizontal:MAIn:SCAle command. It is provided to maintain program compatibility with some older models of Tektronix scopes.

Group

Horizontal

Syntax

HORizontal: SECdiv <NR3>

HORizontal: SECdiv ?

HORizontal:MAIn:SECdiv <NR3>

HORizontal:MAIn:SECdiv?

Arguments

<NR3> specifies the range from 2 ns to 100 s, depending on the model.

Examples

HORizontal:MAIn:SECdiv 2E-6 sets the main scale to 2 µs per division.

HORizontal:MAIn:SECdiv? might return 2.0000E-06 indicating that the main scale is currently set to 2 µs per division.

HORizontal:MAIn:UNIts[:STRing]

Returns the current horizontal unit "s". Query only.

Group

Horizontal

Syntax

HORizontal:MAIn:UNIts?

HORizontal:MAIn:UNIts:STRing?

Examples

HORizontal:MAIn:UNIts? might return SECONDS.

HORizontal:PREViewstate

Returns a boolean value to indicate whether the acquisition system is in the preview state. Query only.

Group

Horizontal

Syntax

HORizontal:PREViewstate?

Returns

<NR1> = 1 if the acquisition system is in the preview state.

<NR1> = 0 if the acquisition system is not in the preview state.

Examples

HORizontal:PREViewstate? might return 1 indicating the acquisition system is in the preview state.

HORizontal:RECOrdlength

Sets the horizontal record length of acquired waveforms. The query form of this command returns the current horizontal record length.

Group

Horizontal

Syntax

`HORizontal:RECOrdlength <NR1>`

`HORizontal:RECOrdlength?`

Arguments

`<NR1>` represents the supported values for horizontal record lengths, which are: 1000, 2000, 20000

Examples

`HORIZONTAL:RECORDLENGTH 2000` specifies that 2000 data points will be acquired for each record.

`HORIZONTAL:RECORDLENGTH?` might return `2000` indicating that the horizontal record length is equal to 2000 data points.

HORizontal:RECOrdlength:Auto

Sets or queries the horizontal record length mode.

Group

Horizontal

Syntax

`HORizontal:RECOrdlength:Auto <NR1>`

`HORizontal:RECOrdlength:Auto?`

Arguments

<NR1> sets the record length mode. 1 enables auto record length mode, 0 disables auto record length mode.

Examples

`HORizontal:RECOrdlength:Auto 1` enables auto record length mode of the analog channels.

`HORizontal:RECOrdlength:Auto?` might return `0` indicating that auto record length mode of the analog channels is off.

HORizontal:RESOlution

Sets or returns the horizontal record length of acquired waveforms. The sample rate is automatically adjusted at the same time to maintain a constant time per division. The query form of this command returns the current horizontal record length.

Group

Horizontal

Syntax

HORizontal:RESOlution <NR1>

HORizontal:RESOlution?

Arguments

<NR1> represents the supported values for horizontal record lengths.

Examples

HORizontal:RESOlution 20000 set the record length to 20000 points.

HORizontal:ROLL

Returns the current horizontal roll mode state: on/off. Query only.

Group

Horizontal

Syntax

HORizontal:ROLL?

Examples

HORizontal:ROLL? might return HORizontal:ROLL ON indicating that roll mode is on.

HORizontal:TRIGger:POSition

Sets or queries the horizontal position when delay mode is OFF. It is similar to HORizontal:POSition.

Sets the

Group

Horizontal

Syntax

HORizontal:TRIGger:POSition <NR3>

HORizontal:TRIGger:POSition?

Arguments

<NR3> is the horizontal position expressed as the percentage of the waveform displayed left of the center of the graticule.

Examples

HORizontal:MAIn:POSition 50 sets the horizontal position to 50%.

HORizontal:MAIn:POSition? might return 100, indicating that the horizontal position is set to 100%.

I commands

This section lists commands and queries that begin with the letter I.

ID?

Returns identifying information about the instrument and its firmware in Tektronix Codes and Formats notation. Query only.

Note: ID? must be the last command when part of a concatenated statement. Otherwise the instrument generates event message 440.

The ID? and *IDN? responses are slightly different.

Group

Miscellaneous

Syntax

```
ID?
```

Returns

Returns the instrument identification in the following format for TBS1000C instruments:

ID TEK/<model number>,CF:91.1CT FV:v<instrument firmware version number>

Examples

`ID?` might return the following response `ID TEK/TBS2104,CF:91.1CT,FV:v2015-12-10_01-00-59rootfs; FPGA:v1.21;`

*IDN?

Returns the instrument identification code in IEEE 488.2 notation. Query only.

Note: *IDN? must be the last command when part of a concatenated statement. Otherwise the instrument generates event message 440.

The *IDN? and ID? responses are slightly different.

Group

Miscellaneous

Syntax

```
*IDN?
```

Returns

Returns the instrument identification in the following format for TBS1000C instruments:

TEKTRONIX,<model number>,CF:91.1CT FV:v<instrument firmware version number> TBS 1XXXC:v<module firmware version number>

Examples

*IDN? might return the following response for a TBS 1072C instrument with the serial number CU10100: TEKTRONIX,TBS 1072C,CU10100,CF:91.1CT FV:v2015-12-10_01-00-59rootfs; FPGA:v1.21;

L commands

This section lists commands and queries that begin with the letter L.

LANGuage

Sets or queries the languages that the instrument uses to display information on the screen. This is equivalent to setting the Language option in the Utility menu.

Group

Miscellaneous

Syntax

`LANGuage`

`LANGuage?`

Arguments

Specifies the language used to display instrument information on the screen.

Examples

`LANGuage FRENch` specifies that the instrument displays information in French.

`LANGuage?` might return `SPANISH`.

LOCk

Enables and disables all front-panel buttons and knobs. There is no front-panel equivalent.

When the front panel is locked, neither the FPAnel:PRESS nor the FPAnel:TURN commands work. They will not generate an error event either. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands. For example, to set the trigger level to 50%, you could use TRIGger:A SETLevel. To force a trigger, you could use TRIGger FORCe.

Group

Miscellaneous

Syntax

`LOCk {ALL|NONe}`

`LOCk?`

Related commands

UNLock on page 161

Arguments

`ALL` disables all front-panel controls.

`NONe` enables all front-panel controls. This is equivalent to the UNLock `ALL` command.

Examples

`LOCk ALL` locks the front-panel controls.

`LOCk?` might return `:LOCK NONE` indicating the front-panel controls are enabled by this command.

*LRN?

This is identical to the query. Query only.

Miscellaneous

Group

Miscellaneous

Syntax

`*LRN?`

M commands

This section lists commands and queries that begin with the letter M.

MATH?

Returns the current math parameters. Query only.

Group

Math

Syntax

```
MATH?
```

Related commands

MATH:DEFINE on page 104, *MATH:VERtical:SCAle* on page 107, *MATH:VERtical:POSition* on page 107, *MATH:VERTical:UNIts* on page 108, *MATH:HORizontal:SCALe* on page 105, *MATH:HORizontal:POSition* on page 105, *MATH:HORizontal:UNIts* on page 106

Returns

Returns the current math parameters:

- The definition of the math waveform:
- Source1 operation source2
- Vertical scale
- Vertical position
- Vertical units
- Horizontal scale
- Horizontal position
- Horizontal units

Examples

`MATH?` might return `"CH1+CH2";2.000;0.0E+0"V",20.0000E-6;0.0E0;"s"`.

MATH:DEFINE

Sets or returns the math waveform definition for the active math operation.

Note: Remember that <QString> must be enclosed in quotes. You can use white space characters between words.

Group

Math

Syntax

```
MATH:DEFINE <QString>
```

MATH:DEFINE?

Arguments

<QString> specifies a math waveform, and can be one of the following,

CH1+CH2, CH1-CH2, CH2-CH1,

CH1*CH2

Examples

MATH:DEFINE "CH1-CH2" sets the math waveform so that it displays the difference of channel 1 and channel 2.

MATH:DEFine? Might return "CH1-CH2".

MATH:HORizontal:POSition

Sets or queries the math horizontal display position for math waveforms. The horizontal position of a dual math waveform with a channel waveform source is set through the commands described in the horizontal section.

Group

Math

Syntax

MATH:HORizontal:POSition <NR2>

MATH:HORizontal:POSition?

Arguments

<NR2> is the math hhorizontal position in percent of record.

Examples

MATH:HORizontal:POSition 20 sets the math horizontal position to 20% of the record.

MATH:HORizontal:POSition? might return 20.000 indicating that the math horizontal position is at 20% of the record.

MATH:HORizontal:SCALe

Sets or queries the math horizontal display scale for dual math waveforms that only have source waveforms. The horizontal scale of a dual math waveform with a channel source waveform is set through the HORizontal:SCALe command.

Group

Math

Syntax

MATH:HORizontal:SCALe <NR3>

MATH:HORizontal:SCALe?

Arguments

<NR3> is the math display scale.

Examples

MATH:HORizontal:SCALe 20.000E-6 sets the math horizontal display scale to 20 µs per division.

MATH:HORizontal:SCALe? might return 20.000E-6 indicating the math horizontal display scale is 20 µs per division.

MATH:HORizontal:UNIts

Queries the math horizontal measurement units label.

Group

Math

Syntax

MATH:HORizontal:UNIts?

Arguments

<Qstring> is a quoated string representing the math horizontal units.

Examples

MATH:HORizontal:UNIts? might return "us" indicating the math horizontal units are µs.

MATH:LABel

This command sets or queries the waveform label for the math waveform.

Group

Math

Syntax

MATH:LABel <Qstring>

MATH:LABel?

Arguments

<Qstring> is the quoted string used as the label for the math waveform.

Examples

MATH:LABEL Output sets the label for the math waveform to "Output."

MATH:LABEL? might return MATH:LABEL "Sum of channel 1 and channel 2" indicating the current label for the math waveform.

MATH:VERtical:POSition

Sets or queries the math waveform display position.

Group

Math

Syntax

`MATH:VERtical:POSition <NR3>`

`MATH:VERtical:POSition?`

Arguments

`<NR3>` specifies the math vertical position in divisions from center screen.

Examples

`MATH:VERtical:POSition 4` sets the math vertical position to 4 divisions above center screen.

`MATH:VERtical:POSition?` might return `-3.000`, indicating that the math waveform is 3 divisions below center screen.

MATH:VERtical:SCAle

Sets or queries the vertical display scale, which should not be confused with the math waveform vertical scale returned in the math waveform pre–amble (MATH?). The display scale is the same as that adjusted through the instrument vertical scale knob that controls waveform zoom factors. The math waveform scale is not affected by this control, rather the math calculation software automatically determines the optimum vertical scale through examination of input waveform data.

Note: The vertical display scale is reset to the waveform pre–amble scale whenever a vertical scale change to a math source waveform results in a new math autoscale operation. The vertical display scale should be changed only after math source waveform adjustments are complete.

Group

Math

Syntax

`MATH:VERtical:SCAle <NR3>`

`MATH:VERtical:SCAle?`

Arguments

`<NR3>` specifies the math vertical scale in units per division.

Examples

`MATH:VERtical:SCAle 5.0E0` sets the math vertical scale to five math waveform units per division.

`MATH:VERTical:SCALe?` might return 5.000.

MATH:VERTical:UNIts

Queries the math vertical measurement units.

Group

Math

Syntax

MATH:VERTical:UNIts?

Examples

`MATH:VERTical:UNIts?` might return `"V"`.

MEASUrement?

Returns the current MEASUrement settings. Query only.

Group

Measurement

Syntax

`MEASUrement?`

Returns

Instrument measurement settings.

Examples

`MEASUrement?` might return the following:

```
:MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2
RISE;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2
CH2;:MEASUREMENT:MEAS1:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2
RISE;:MEASUREMENT:MEAS1:STATE 1;TYPE FREQUENCY;UNITS "Hz";SOURCE1
CH1;SOURCE2 CH2;COUNT 0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0
000;STDDEV 0.0000;:MEASUREMENT:MEAS2:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2
RISE;:MEASUREMENT:MEAS2:STATE 1;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2;COUNT
0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0000;STDDEV 0.0000;:MEASUREMENT:MEAS3:
DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:MEAS3:STATE 1;TYPE
PK2PK;UNITS "V";SOURCE1 CH1;SOURCE2 CH2;COUNT 0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM
0.0000;STDDEV 0.0000;:MEASUREMENT:MEAS4:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2
RISE;:MEASUREMENT:MEAS4:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2;COUNT
0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0000;STDDEV 0.0000;:MEASUREMENT:METHOD
AUTO;REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0000;LOW 0.0000;MID1 0.0000;MID2
0.0000;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0000;LOW 10.0000;MID1 50.0000;MID2
50.0000;:MEASUREMENT:INDICATORS:STATE OFF;NUMHORZ 0;NUMVERT 0;HORZ1 99.0000E
+36;HORZ2 99.0000E+36;HORZ3 99.0000E+36;HORZ4 99.0000E+36;VERT1 99.0000E+36;VERT2
```

```
99.0000E+36;VERT3 99.0000E+36;VERT4 99.0000E+36;:MEASUREMENT:STATISTICS:MODE
OFF;WEIGHTING 32;:MEASUREMENT:GATING SCREEN.
```

MEASUrement:CLEARSNapshot

Clears the existing measurement snapshot results and removes the snapshot window. Command only, no query form.

Group

Measurement

Syntax

`MEASUrement:CLEARSNapshot`

Examples

`MEASUrement:CLEARSNapshot` clears the existing snapshop results and removes the snapshot window.

MEASUrement:ENABLE

Allows educators to disable or enable the Measurement functions. The function can be manually set from the Utility menu. To access the menu, refer to the product user manual.

Group

Miscellaneous

Syntax

`MEASUrement:ENABLE {ON|OFF}`

`MEASUrement:ENABLE?`

Related Commands

MEASUrement? on page 108

FEAEN:PASSWORD on page 73

Arguments

`ON` enables the measurement feature.

`OFF` disables the measurement feature.

Examples

`MEASUrement:ENABLE OFF` disable measurement.

`MEASUrement:ENABLE?` might return `1` indicating that measurements is enabled.

MEASUrement:GATing

Sets or queries the measurement gating setting.

Group

Measurement

Syntax

`MEASUrement:GATing {OFF|SCREen|CURSor}`

`MEASUrement:GATing?`

Arguments

`OFF` turns off measurement gating (full record).

`SCREen` turns on gating, using the left and right edges of the screen.

`CURSor` limits measurements to the portion of the waveform between the vertical bar cursors, even if they are off screen.

Examples

`MEASUREMENT:GATING CURSOR` turns on measurement gating using the cursors as limits.

`MEASUREMENT:GATING?` might return `:MEASUREMENT:GATING CURSOR` indicating that measurements are limited to the portion of the waveform between the vertical bar cursors.

MEASUrement:IMMed?

Returns all immediate measurement setup parameters. Immediate queries and commands are the preferred methods for programming. An immediate measurement selection is not visible or accessible through the display screen or front panel. Query only.

Group

Measurement

Syntax

`MEASUrement:IMMed?`

Returns

Immediate measurement setup parameters

Examples

`MEASUrement:IMMed?` might return `:MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2.`

MEASUrement:IMMed:DELay?

Returns information about the immediate delay measurement. This command is equivalent to viewing the delay measurement settings on the measurement readout. Query only.

Group

Measurement

Syntax

`MEASUrement:IMMed:DELay?`

Examples

`MEASUREMENT:IMMED:DELAY?` might return `:MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS; EDGE1 RISE; EDGE2 RISE`.

MEASUrement:IMMed:DELay:EDGE<x>

Sets or queries the slope of the edge the instrument uses for the delay from or to waveform when taking an immediate delay measurement.

Group

Measurement

Syntax

`MEASUrement:IMMed:DELay:EDGE<x> {FALL|RISe}`

`MEASUrement:IMMed:DELay:EDGE<x>?`

Arguments

`<x>` specifies which waveform to use, where <x> = 1 is the from waveform, and <x> = 2 is the to waveform.

`FALL` specifies the falling edge.

`RISe` specifies the rising edge.

Examples

`MEASUREMENT:IMMED:DELAY:EDGE1 RISE` specifies that the from waveform rising edge be used for the immediate delay measurement.

`MEASUREMENT:IMMED:DELAY:EDGE1?` returns either RISE or FALL.

MEASUrement:IMMed:SOUrce1

Sets or queries the source for all single source immediate measurements and specifies the source to measure from when taking an immediate delay or phase measurement.

Note: If you do not specify a numerical suffix, the source is assumed to be SOURCE 1.

Group

Measurement

Syntax

MEASUrement:IMMed:SOUrce1 {CH<x> | MATH}

MEASUrement:IMMed:SOUrce1?

Related commands

MEASUrement:IMMed:SOUrce2 on page 112

Arguments

CH<x> specifies the measurement source channel as one of the input channels. The value of <x> can vary from 1 through 2.

MATH specifies the measurement source channel as the math waveform.

Examples

MEASUREMENT:IMMED:SOURCE1 CH1 specifies channel 1 as the immediate measurement source.

MEASUREMENT:IMMED:SOURCE1? might return CH2 indicating that channel 2 is the immediate measurement source.

MEASUrement:IMMed:SOUrce2

Sets or queries the secondary source for dual-source immediate measurements.

Note: Source2 measurements only apply to phase and delay measurement types, which require both a target (Source1) and reference (Source2) source.

Group

Measurement

Syntax

MEASUrement:IMMed:SOUrce2 {CH<x> | MATH}

MEASUrement:IMMed:SOUrce2?

Related commands

MEASUrement:IMMed:SOUrce1 on page 111

Arguments

CH<x> specifies the measurement source channel as one of the input channels. The value of <x> can vary from 1 through 2.

MATH specifies the measurement source channel as the math waveform.

Examples

MEASUrement:IMMed:SOUrce2 CH2 sets the immediate measurement source 2 to channel 2.

MEASUrement:IMMed:SOUrce2? might return MATH indicating that Math is the immediate measurement source.

MEASUrement:IMMed:TYPe

Sets or queries the immediate measurement type.

Group

Measurement

Syntax

`MEASUrement:IMMed:TYPe {AMPlitude|AREa|BURst|CAREa|CMEan|CRMs|DELay|FALL|FREQuency|HIGH|LOW|MAXimum|MEAN|MINImum|NDUty|NEDGECount|NOVershoot |NPULSECount|NWIdth|PEDGECount|PDUty |PERIod|PHAse|PK2Pk|POVershoot|PPULSECount|PWIdth|RISe|RMS}`

`MEASUrement:IMMed:TYPe?`

Arguments

`AMPlitude` measures the amplitude of the selected waveform. It measures the high value less the low value measured over the entire waveform or gated region. Amplitude = High – Low

`AREa` measures the voltage over time. The area is over the entire waveform or gated region and is measured in volt-seconds. The area measured above the ground is positive, while the area below ground is negative.

`BURst` measures the duration of a burst. The measurement is made over the entire waveform or gated region.

`CAREa` (cycle area) measures the voltage over time. It measures, in volt-seconds, the area over the first cycle in the waveform or the first cycle in the gated region. The area measured above the common reference point is positive, while the area below the common reference point is negative.

`CMEan` (cycle mean) measures the arithmetic mean over the first cycle in the waveform or the first cycle in the gated region.

`CRMs` (cycle rms) measures the true Root Mean Square voltage over the first cycle in the waveform or the first cycle in the gated region.

`DELay` measures the time between the middle reference (default = 50%) amplitude point of the source waveform and the destination waveform.

`FALL` measures the time taken for the falling edge of the first pulse in the waveform or gated region to fall from a high reference value (default is 90%) to a low reference value (default is 10%).

`FREQuency` measures the first cycle in the waveform or gated region. Frequency is the reciprocal of the period and is measured in hertz (Hz), where 1 Hz = 1 cycle per second.

`HIGH` measures the High reference (100% level, sometimes called Topline) of a waveform.

`LOW` measures the Low reference (0% level, sometimes called Baseline) of a waveform.

`MAXimum` finds the maximum amplitude. This value is the most positive peak voltage found. It is measured over the entire waveform or gated region.

`MEAN` amplitude measurement finds the arithmetic mean over the entire waveform or gated region.

`MINImum` finds the minimum amplitude. This value is typically the most negative peak voltage. It is measured over the entire waveform or gated region.

`NDUty` (negative duty cycle) is the ratio of the negative pulse width to the signal period, expressed as a percentage. The duty cycle is measured on the first cycle in the waveform or gated region. Negative Duty Cycle = ((Negative Width) / Period) × 100%

`NEDGECount` is the count of falling edges.

`NOVershoot` (negative overshoot) finds the negative overshoot value over the entire waveform or gated region. Negative Overshoot = ((Low – Minimum) / Amplitude) × 100%)

NPULSECount is the count of negative pulses.

NWIdth (negative width) measurement is the distance (time) between the middle reference (default = 50%) amplitude points of a negative pulse.

PDUty (positive duty cycle) is the ratio of the positive pulse width to the signal period, expressed as a percentage. It is measured on the first cycle in the waveform or gated region. Positive Duty Cycle = ((Positive Width / Period) × 100%

PEDGECount is the count of rising edges.

PERIod is the time required to complete the first cycle in a waveform or gated region. Period is the reciprocal of frequency and is measured in seconds.

PHAse measures the phase difference (amount of time a waveform leads or lags the reference waveform) between two waveforms. The measurement is made between the middle reference points of the two waveforms and is expressed in degrees, where 360° represents one waveform cycle.

PK2Pk (peak-to-peak) finds the absolute difference between the maximum and minimum amplitude in the entire waveform or gated region.

POVershoot is the positive overshoot value over the entire waveform or gated region. Positive Overshoot = ((Maximum – High) / Amplitude) ×100%

PPULSECount is the count of positive pulses.

PWIdth (positive width) is the distance (time) between the middle reference (default = 50%) amplitude points of a positive pulse. The measurement is made on the first pulse in the waveform or gated region.

RISe timing measurement finds the rise time of the waveform. The rise time is the time it takes for the leading edge of the first pulse encountered to rise from a low reference value (default is 10%) to a high reference value (default is 90%).

RMS amplitude measurement finds the true Root Mean Square voltage in the entire waveform.

Examples

MEASUrement:IMMed:TYPe FREQUENCY defines the immediate measurement to be a frequency measurement.

MEASUREMENT:IMMED:TYPE? might return RMS indicating that the immediate measurement is the true Root Mean Square voltage.

MEASUrement:IMMed:UNIts?

Returns the units for the immediate instrument measurement. Query only.

Group

Measurement

Syntax

MEASUrement:IMMed:UNIts?

Returns

<QString> where the string is one of the following: VOLTS, VOLTS SQUARED, SEC, HERTZ, PERCENT, DIVS, SAMPLES, OHMS, AMPS, WATTS, MINUTES, DEGREES, UNKNOWN, AMPS SQUARED, HOURS, DAYS, DB, BYTES, INVERSE HERTZ, IRE, V OVER V, V OVER A, VOLTS WATTS, V OVER W, VOLTS DB, V OVER DB, A OVER V, A OVER A, AMPS WATTS, A OVER W, AMPS DB, A OVER DB, WATTS VOLTS, W OVER V, WATTS AMPS, W OVER A, WATTS SQUARED, W OVER W, WATTS DB, W OVER DB,

DB VOLTS, DB OVER V, DB AMPS, DB OVER A, DB WATTS, DB OVER W, DB SQUARED, DB OVER DB, VOLTS SEC, AMPS SEC, WATTS SEC, V OVER S, A OVER S, W OVER S.

Examples

`MEASUrement:IMMed:UNIts?` might return `:MEASUREMENT:IMMED:UNIts "s"`, indicating that the unit for the immediate measurement is seconds.

MEASUrement:IMMed:VALue?

Returns the value of the measurement specified by the `MEASUrement:IMMed:TYPe` command. The measurement is immediately taken on the source(s) specified by a `MEASUrement:IMMed:SOUrce1` command. Query only.

Note: A change to HORizontal:MAIn:SCALe or CH<x>:SCALe will not necessarily have taken affect if immediately followed by this command.

Group

Measurement

Syntax

`MEASUrement:IMMed:VALue?`

Related Commands

MEASUrement:IMMed:TYPe on page 113,

**ESR?* on page 70, *EVENT?* on page 70

Returns

`<NR3>`

Examples

`MEASUrement:IMMed:VALue?` might return `28.75E6` if you are measuring the frequency of a 28.75 MHz signal.

`MEASUrement:IMMed:VALue?` might return `9.9000E+37`. If the measurement has an error or warning associated with it, then an item is added to the error queue. The error can be checked for with the *ESR? and ALLEv? commands..

MEASUrement:MEAS<x>?

Returns all measurement parameters for the displayed instrument periodic measurement specified by <x>. Where <x> identifies the measurement, 1 through 6 depending on instrument model. Query only.

Group

Measurement

Syntax

`MEASUrement:MEAS<x>?`

Returns

Settings for the specified measurement source.

Examples

`MEASUREMENT:MEAS3?` might return `PERIOD;"s";CH1`

MEASUrement:MEAS<x>:DELay?

Returns the delay measurement parameters for the measurement specified by <x>, which ranges from 1 through 6. Query only.

Group

Measurement

Syntax

`MEASUrement:MEAS<x>:DELay?`

Examples

`MEASUrement:MEAS1:DELay?` might return `:MEASUREMENT:MEAS1:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE`.

MEASUrement:MEAS<x>:DELay:EDGE<x>

Sets or queries the slope of the edge used for the delay from or to waveform when taking an immediate delay measurement. The waveform is specified by `MEASUrement:MEAS<x>:SOURCE[1]`.

Group

Measurement

Syntax

`MEASUrement:MEAS<x>:DELay:EDGE<x> {FALL|RISe}`

`MEASUrement:MEAS<x>:DELay:EDGE<x>?`

Arguments

`<x>` specifies which waveform to use, where <x> = 1 is the "from" waveform, and <x> = 2 is the "to" waveform.

`FALL` specifies the falling edge.

`RISe` specifies the rising edge.

Examples

`MEASUREMENT:MEAS1:DELAY:EDGE1 RISE` specifies that the "from" waveform rising edge be used for the immediate delay measurement.

`MEASUREMENT:MEAS1:DELAY:EDGE1?` returns either RISE or FALL

MEASUrement:MEAS<x>:SOUrce1

Sets or queries the source for all single source measurements and specifies the source to measure from when taking a delay measurement or phase measurement. Where <x> identifies the measurement, 1 through 6. This is equivalent to selecting the measurement source in the MEASURE menu.

Group

Measurement

Syntax

`MEASUrement:MEAS<x>:SOUrce1 {CH<x>|MATH}`

`MEASUrement:MEAS<x>:SOUrce1?`

Arguments

`CH<x>` specifies the input channel source for the measurement.

`MATH` specifies the measurement source channel as the math waveform.

Examples

`MEASUREMENT:MEAS2:SOURCE1 CH1` sets source1 for Measurement 2 to channel 1.

`MEASUrement:MEAS1:SOUrce1?` might return `MATH` indicating the source for measurement 1 is the math waveform.

MEASUrement:MEAS<x>:SOUrce2

For SOURce1: Sets or queries the source for all single channel measurements. For delay or phase measurements, sets or queries the waveform to measure from. For SOUrce2: Sets or queries the waveform to measure to when taking a delay measurement or phase measurement (two-source waveforms measurements).

Group

Measurement

Syntax

`MEASUrement:MEAS<x>:SOUrce2 {CH<x>|MATH}`

`MEASUrement:MEAS<x>:SOUrce2?`

Related commands

MEASUrement:MEAS<x>:TYPe on page 118

Arguments

`CH<x>` specifies the input channel source for the measurement, where x is the channel number.

`MATH` specifies the measurement source channel as the math waveform.

Examples

`MEASUrement:MEAS1:SOUrce2 CH1` sets source2 for Measurement 2 to channel 1.

MEASUrement:MEAS1:SOUrce2? might return MATH indicating the to source for measurement 1 is the math waveform.

MEASUrement:MEAS<x>:STATE

Sets or queries whether the specified measurement slot is computed and displayed. The measurement slot is specified by <x>, which ranges from 1 through 6. For a measurement to display, you must have selected a source waveform and defined the measurement you want to take and display. You select the measurement using the MEASUrement:MEAS<x>:SOURCE[1] command. You define the measurement type using the MEASUrement:MEAS<x>:TYPe command.

Group

Measurement

Syntax

MEASUrement:MEAS<x>:STATE {OFF|ON|<NR1>}

MEASUrement:MEAS<x>:STATE?

Related commands

MEASUrement:MEAS<x>:TYPe on page 118

MEASUrement:MEAS<x>:SOUrce1 on page 117

Arguments

OFF disables calculation and display of the specified measurement slot.

ON enables calculation and display of the specified measurement slot.

<NR1> = 0 disables calculation and display of the specified measurement slot; any other value enables calculation and display of the specified measurement slot.

Examples

MEASUrement:MEAS2:STATE ON computes and displays the measurement defined as measurement 2.

MEASUrement:MEAS1:STATE? might return :MEASUREMENT:MEAS1:STATE 0 indicating that measurement defined for measurement slot 1 is disabled.

MEASUrement:MEAS<x>:TYPe

Sets or queries the on-screen periodic instrument measurement type for the measurement specified by <x>. Where <x> identifies the measurement, 1 through 6 depending on instrument model.

This is equivalent to selecting the measurement type in the MEASURE menu. Setting the type to anything other than NONE displays the MEASURE menu on the screen.

Note: You should use the MEASUrement:IMMed command with programming to take measurements, as this is preferred to the MEASUrement:MEAS<x>command.

Group

Measurement

Syntax

```
MEASUrement:MEAS<x>:TYPe {AMPlitude | AREa | BURst | CARea | CMEan | CRMs | DELay
| FALL |FREQuency | HIGH | LOW | MAXimum | MEAN | MINImum | NDUty | NEDGECount
| NOVershoot | NPULSECount|NWIdth | PDUty | PEDGECount | PERIod | PHAse | PK2Pk |
POVershoot | PPULSECount | PWIdth | RISe | RMS}
```

```
MEASUrement:MEAS<x>:TYPe?
```

Arguments

`AMplitude` is the high value less the low value measured over the entire waveform or gated region.

Amplitude = High – Low.

`AREA` is a voltage over time measurement. It returns the area over the entire waveform or gated region in volt-seconds. Area measured above ground is positive. Area measured below ground is negative.

`BURst` measures the duration of a burst. The measurement is made over the entire waveform or gated region.

`CAREA` (cycle area) is a voltage over time measurement. The measurement is the area over the first cycle in the waveform or the first cycle in the gated region expressed in volt-seconds. The area above the common reference point is positive. The area below the common reference point is negative.

`CMEAN` (cycle mean) is the arithmetic mean over the first cycle in the waveform or the first cycle in the gated region.

`CRMs` is the true Root Mean Square voltage of the first complete cycle in the waveform.

`DELay` measures the time between the middle reference (default = 50%) amplitude point of the source waveform and the destination waveform.

`FALL` is the fall time between 90% and 10% (defaults) of the first falling edge of the waveform or gated region. Falling edge must be displayed to measure. The instrument automatically calculates the 10% and 90% measurement points.

`FREQuency` measures the first cycle in the waveform or gated region. Frequency is the reciprocal of the period and is measured in hertz (Hz), where 1 Hz = 1 cycle per second.

`HIGH` is the value used as 100% whenever high reference, mid reference, or low reference values are needed, such as in fall time or rise time measurements. Calculate using either the min-max or histogram method. The min-max method uses the maximum value found. The histogram method uses the most common value found above the midpoint. This value is measured over the entire waveform or gated region.

`LOW` is the value used as 0% whenever high reference, mid reference, or low reference values are needed, such as in fall time or rise time measurements. Calculate using either the min-max or histogram method. The min-max method uses the minimum value found. The histogram method uses the most common value found below the midpoint. This value is measured over the entire waveform or gated region.

`MAXImum` finds the maximum amplitude. This value is the most positive peak voltage found. It is measured over the entire waveform or gated region.

`MEAN` is the arithmetic mean over the entire waveform or gated region.

`MINImum` finds the minimum amplitude. This value is typically the most negative peak voltage. It is measured over the entire waveform or gated region.

`NDUty` is the ratio of the negative pulse width to the signal period expressed as a percentage. The duty cycle is measured on the first cycle in the waveform or gated region.

Negative Duty Cycle = ((Negative Width) / Period) × 100%

`NEDGECount` is the number of negative transitions from the high reference value to the low reference value in the waveform or gated region.

NOVERshoot is measured over the entire waveform or gated region and is expressed as:

Negative Overshoot = (Low – Minimum) / Amplitude * 100%.

NPULSECount is the number of negative pulses that fall below the mid reference crossing in the waveform or gated region.

NWIdth (negative width) measurement is the distance (time) between the middle reference (default = 50%) amplitude points of a negative pulse.

PDUty (positive duty cycle) is the ratio of the pulse width to the signal period, expressed as a percentage. It is measured on the first cycle in the waveform or gated region.

Positive Duty Cycle = ((Positive Width) / Period) × 100%

PEDGECount is the number of positive transitions from the low reference value to the high reference value in the waveform or gated region.

PERIod is the duration, in seconds, of the first complete cycle in the waveform or gated region. Period is the reciprocal of frequency and is measured in seconds.

PHAse measures the phase difference (amount of time a waveform leads or lags the reference waveform) between two waveforms. The measurement is made between the middle reference points of the two waveforms and is expressed in degrees, where 360° represents one waveform cycle.

PK2pk (peak-to-peak) finds the absolute difference between the maximum and minimum amplitude in the entire waveform or gated region.

POVERshoot is the positive overshoot value over the entire waveform or gated region. The measurement is expressed as:

Positive Overshoot = (Maximum – High) / Amplitude * 100%.

PPULSECount is the number of positive pulses that rise above the mid reference crossing in the waveform or gated region.

PWIdth (positive width) is the distance (time) between the first rising edge and the next falling edge at the waveform 50% level (default). Rising and falling edges must be displayed to measure. The measurement is made on the first pulse in the waveform or gated region. The instrument automatically calculates the 50% measurement point.

RISe is the rise time between 10% and 90% of the first rising edge of the waveform or gated region. Rising edge must be displayed to measure. The instrument automatically calculates the 10% and 90% measurement points.

RMS amplitude measurement finds the true Root Mean Square voltage in the entire waveform or gated region.

Examples

MEASUREMENT:MEAS2:TYPE FREQUENCY specifies MEAS2 to measure the frequency of a waveform.

MEASUREMENT:MEAS2:TYPE? might return :MEASUREMENT:MEAS1:TYPE RMS indicating that measurement 1 is defined to measure the RMS value of a waveform.

MEASUrement:MEAS<x>:UNIts?

Returns the units for the instrument measurement specified by MEASUrement:MEAS<x>:TYPe. Where <x> identifies the measurement, 1 through 6. Query only.

Group

Measurement

Syntax

MEASUrement:MEAS<x>:UNIts?

Related commands

MEASUrement:MEAS<x>:TYPe on page 118,

MEASUrement:IMMed:UNIts? on page 114

Returns

<QString> returns the units for the measurement.

Examples

MEASUREMENT:MEAS3:UNITS? might return :MEASUREMENT:MEAS1:UNIts % indicating units for measurement 1 are set to percent.

MEASUrement:MEAS<x>:VALue?

Returns the value that was calculated for the instrument on-screen periodic measurement specified by <x>. Where <x> identifies the measurement, 1 through 6. Query only.

This is the same value as displayed on-screen. If measurement statistics are enabled, a new value is calculated with every waveform. In addition, this value is updated approximately every 1/3 second. If you are acquiring a long acquisition record, the instrument may take longer to update.

Group

Measurement

Syntax

MEASUrement:MEAS<x>:VALue?

Related commands

MEASUrement:MEAS<x>:UNIts? on page 120

**ESR?* on page 70

ALLEv? on page 29

Returns

<NR3> is the measurement value.

Examples

MEASUREMENT:MEAS3:VALUE? might return 28.75E6 if measurement number three is frequency. If the measurement has an error or warning associated with it, then an item is added to the error queue. The error can be checked for with the *ESR? and ALLEv? commands.

MEASUrement:REFLevel?

Returns the current reference level parameters. Query only.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

```
MEASUrement:REFLevel?
```

Example

`MEASUREMENT:REFLEVEL?` might return these reference level settings `PERCENT;ABSOLUTE:HIGH 0.0000;LOW 0.0000;MID1 0.0000;MID2 0.0000;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0000;LOW 10.0000;MID1 50.0000; MID2 50.0000`

MEASUrement:REFLevel:ABSolute:LOW

Sets or returns the low reference level, and is the lower reference level when MEASUrement:REFLevel:METHod is set to Absolute.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

```
MEASUrement:REFLevel:ABSolute:LOW <NR3>
```

```
MEASUrement:REFLevel:ABSolute:LOW?
```

Related commands

MEASUrement:REFLevel:METHod on page 124

MEASUrement:IMMed:TYPe on page 113

MEASUrement:MEAS<x>:TYPe on page 118

Arguments

`<NR3>` is the low reference level, in volts. The default is 0.0 V.

Example

`MEASUREMENT:REFLEVEL?` might return these reference level settings:`MEASUREMENT:REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0000;LOW 0.0000;MID1 0.0000;MID2 0.0000;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0000;LOW 10.0000;MID1 50.0000; MID2 50.0000`

MEASUrement:REFLevel:ABSolute:MID1

Sets or returns the mid reference level, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to Absolute. This command affects the results of period, frequency, delay, and all cyclic measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

`MEASUrement:REFLevel:ABSolute:MID[1] <NR3>`

`MEASUrement:REFLevel:ABSolute:MID[1]?`

Related commands

MEASUrement:REFLevel:METHod on page 124

Arguments

`<NR3>` is the mid reference level, in volts. The default is 0.0 V.

Example

`MEASUREMENT:REFLEVEL:ABSOLUTE:MID 1.71` sets the mid reference level to 0.71 V.

`MEASUREMENT:REFLEVEL:ABSOLUTE:MID?` might return `0.7100E+00` indicating that the absolute mid1 reference level is set to 0.71 V.

MEASUrement:REFLevel:ABSolute:MID2

Sets or returns the mid reference level for the ""to"" waveform when taking a delay measurement, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to Absolute. This command affects the results of delay measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

`MEASUrement:REFLevel:ABSolute:MID2 <NR3>`

`MEASUrement:REFLevel:ABSolute:MID2?`

Related commands

MEASUrement:REFLevel:METHod on page 124

Arguments

<NR3> is the mid reference level, in volts. The default is 0.0 V.

Example

MEASUREMENT:REFLEVEL:ABSOLUTE:MID2 0.5 sets the mid reference level for the delay waveform to 0.5 V.

MEASUREMENT:REFLEVEL:ABSOLUTE:MID2? might return 0.5000E+00 indicating that the absolute mid2 reference level is set to 0.5 V.

MEASUrement:REFLevel:METHod

Sets or returns the reference level units used for measurement calculations.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the eight periodic measurements. To change the parameter for individual measurements, use the MEASUrement:MEAS<x>:REFLevel commands.

Group

Measurement

Syntax

MEASUrement:REFLevel:METHod {ABSolute | PERCent}

MEASUrement:REFLevel:METHod?

Arguments

ABSolute specifies that the reference levels are set explicitly using this commands. This method is useful when precise values are required (for example, when designing to published interface specifications, such as RS-232-C).

PERCent specifies that the reference levels are calculated as a percent relative to HIGH and LOW. The percentages are defined using this commands.

Example

MEASUREMENT:REFLEVEL:METHOD ABSOLUTE specifies that explicit user-defined values are used for the reference levels.

MEASUREMENT:REFLEVEL:METHOD? might return PERCENT indicating that the reference level units used are calculated as a percent relative to HIGH and LOW.

MEASUrement:REFLevel:PERCent:HIGH

Sets or returns the percent (where 100% is equal to HIGH) used to calculate the high reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of rise and fall measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

MEASUrement:REFLevel:PERCent:HIGH <NR3>

MEASUrement:REFLevel:PERCent:HIGH?

Related commands

MEASUrement:REFLevel:METHod on page 124

MEASUrement:IMMed:TYPe on page 113

MEASUrement:MEAS<x>:TYPe on page 118

Arguments

<NR3> is the high reference level, ranging from 0 to 100%. The default high reference level is 90%.

Example

MEASUrement:REFLevel:PERCent:HIGH 95 sets the high reference level to 95% of HIGH.

MEASUrement:REFLevel:PERCent:HIGH? might return 90 indicating that the percentage high reference level is set to 90% of HIGH.

MEASUrement:REFLevel:PERCent:LOW

Sets or returns the percent (where 100% is equal to HIGH) used to calculate the low reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of rise and fall measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

MEASUrement:REFLevel:PERCent:LOW <NR3>

MEASUrement:REFLevel:PERCent:LOW?

Related commands

MEASUrement:REFLevel:METHod on page 124

MEASUrement:IMMed:TYPe on page 113

MEASUrement:MEAS<x>:TYPe on page 118

Arguments

<NR3> is the low reference level, ranging from 0 to 100%. The default low reference level is 10%.

Example

MEASUrement:REFLevel:PERCent:LOW 15 sets the high reference level to 15% of HIGH.

`MEASUrement:REFLevel:PERCent:LOW?` might return `10` indicating that the percentage high reference level is set to 10% of HIGH.

MEASUrement:REFLevel:PERCent:MID1

Sets or returns the percent (where 100% is equal to HIGH) that is used to calculate the mid reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of period, frequency, delay, and all cyclic measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

`MEASUrement:REFLevel:PERCent:MID[1] <NR3>`

`MEASUrement:REFLevel:PERCent:MID[1]?`

Related commands

MEASUrement:REFLevel:METHod on page 124

Arguments

`<NR3>` is the mid reference level, ranging from 0 to 100%. The default mid reference level is 50%.

Example

`MEASUrement:REFLevel:PERCent:MID[1] 60` sets the mid reference level to 60% of HIGH.

`MEASUrement:REFLevel:PERCent:MID[1]?` might return `65` indicating that the percentage mid reference level is set to 65% of HIGH.

MEASUrement:REFLevel:PERCent:MID2

Sets or returns the percent (where 100% is equal to HIGH) that is used to calculate the mid reference level for the second waveform specified when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of delay measurements.

Note: This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.

Group

Measurement

Syntax

`MEASUrement:REFLevel:PERCent:MID2 <NR3>`

`MEASUrement:REFLevel:PERCent:MID2?`

Related commands

MEASUrement:REFLevel:METHod on page 124

Arguments

<NR3> is the mid reference level, ranging from 0 to 100%. The default mid reference level is 50%.

Example

MEASUrement:REFLevel:PERCent:MID2 40 sets the mid reference level to 40% of HIGH.

MEASUrement:REFLevel:PERCent:MID2? might return 45 indicating that the percentage mid reference level is set to 45% of HIGH.

MEASUrement:SNAPSHOT

Displays the measurement snapshot list on the instrument screen. Command only, no query form.

Group

Measurement

Syntax

MEASUrement:SNAPSHOT

MEASUrement:SOURCESNAPShot

Sets or returns the snapshot source.

Group

Measurement

Syntax

MEASUrement:SOURCESNAPShot {CH1|CH2|CH3|CH4|MATH}

MEASUrement:SOURCESNAPShot?

Examples

MEASUrement:SOURCESNAPShot CH1 sets the snapshot source to channel 1.

MEASUrement:SOURCESNAPShot? might return CH1 indicating the snapshot source is channel 1.

O commands

This section lists commands and queries that begin with the letter O.

*OPC

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending commands that generate an OPC message are complete. The *OPC? query places the ASCII character "1" into the output queue when all such OPC commands are complete. The *OPC? response is not available to read until all pending operations finish.

The *OPC command allows you to synchronize the operation of the instrument with your application program. See *Synchronization Methods* on page 187.

Certain instrument operations can affect the *OPC response. See *Table 11* on page 188.

Group

Status and Error

Syntax

`*OPC`

`*OPC?`

Related Commands

BUSY? on page 32, **WAI* on page 163

Examples

`*OPC` generates the operation complete message in the SESR at the completion of all pending OPC operations.

`*OPC ?` might return `1` to indicate that all pending OPC operations are finished.

P commands

This section lists commands and queries that begin with the letter P.

*PSC

Sets or queries the power-on status flag that controls the automatic power-on handling of the DESER, SRER, and ESER registers. When *PSC is true, the DESER register is set to 255 and the SRER and ESER registers are set to 0 at power on. When *PSC is false, the current values in the DESER, SRER, and ESER registers are preserved in nonvolatile memory when power is shut off and are restored at power on. Refer to the Status and Events section for more information. Command only, no query form.

Group

Status and Error

Syntax

`*PSC {OFF|ON|NR1>}`

Related Commands

DESE on page 61, **ESE* on page 69, *FACtory* on page 73, **RST* on page 134, **SRE* on page 143

Arguments

`OFF` sets the power-on status clear flag to false.

`ON` sets the power-on status clear flag to true.

`<NR1>` = 0 sets the power-on status clear flag to false, disables the power on clear, and allows the instrument to possibly assert SRQ after power on.

`<NR1>` ≠ 0 sets the power-on status clear flag true. Sending *PSC 1, therefore, enables the power-on status clear and prevents any SRQ assertion after power on.

Examples

`*PSC 0` sets the power-on status clear flag to false.

`*PSC`? might return the value `1`, showing that the power-on status clear flag is set to true.

R commands

This section lists commands and queries that begin with the letter R.

*RCL

Restores the state of the instrument from a copy of its settings stored in memory. (The settings are stored using the *SAV command.) This command is equivalent to RECAll:SETUp, and performs the same function as the Recall Saved Setup item in the front-panel Save/Recall Setup menu. Command only, no query form.

Group

Save and Recall

Syntax

```
*RCL <NR1>
```

Related Commands

FACtory on page 73, **LRN?* on page 103, *RECAll:SETUp* on page 130, **RST* on page 134, **SAV* on page 135, *SAVe:SETUp* on page 137

Arguments

<NR1> is an integer value in the range from 1 to 10, and specifies a setup storage location.

Examples

*RCL 3 restores the instrument from a copy of the settings stored in memory location 3.

RECAll:SETUp

Restores the factory-default instrument settings, user-saved settings from internal nonvolatile memory, or user-saved settings from a file on a USB flash drive. Using the FACTORY argument is equivalent to pushing the DEFAULT SETUP front-panel button. Command only, no query form.

Group

Save and Recall

Syntax

```
RECAll:SETUp {FACtory|<NR1>|<file path>}
```

Related Commands

FACtory on page 73, **RCL* on page 130, **RST* on page 134, **SAV* on page 135, *SAVe:SETUp* on page 137, *FILESystem:CWD* on page 79

Arguments

FACtory selects the factory setup.

<NR1> is a value in the range from 1 to 10, and specifies a setup memory storage location.

<file path> is a quoted string that defines the path and name of the .SET setup file to recall from a USB drive. Input the file path using the form <drive>:/<dir>/<filename>.<extension> and one or more <dir>s are optional. If you do not specify them, the instrument will read the file from the default directory (see FILESystem:CWD). <filename> stands for a filename; the use of wildcard characters in filenames is not supported. Filename extensions are not required, but highly recommended.

Examples

RECAll:SETUp FACTORY recalls the instrument setup to its factory defaults.

RECALL:SETUP 2 recalls the instrument setup from setup storage location 2.

RECALL:SETUP "TEK00000.SET" recalls the setup from the file TEK00000.SET in the current directory (such as “usb0/”).

RECAll:WAVEForm

Recalls a saved waveform file to a reference location. Command only, no query form.

Group

Save and Recall

Syntax

RECAll:WAVEForm <file path>,REF<x>

Related Commands

SAVe:WAVEform on page 138, *FILESystem:CWD* on page 79, *FILESystem?* on page 78

Arguments

<file path> specifies a location of the instrument setup file. <file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>:/<dir>/<filename>.<extension> and one or more <dir>s are optional. If you do not specify them, the instrument will read the file from the default directory (see FILESystem:CWD). <filename> stands for a filename; the use of wildcard characters in filenames is not supported. Filename extensions are not required, but highly recommended.

REF<x> is the instrument reference memory location in which to load the waveform. You must load a saved waveform into a reference memory location before displaying the waveform. Reference memory location values are 1 or 2.

Examples

RECALL:WAVEFORM "tek00000.isf",REF1 recalls the waveform stored in the file named tek00000.isf from the current directory for waveforms to the reference location 1.

REF<x>?

Returns reference waveform data for the channel specified by <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

REF<x>?

Examples

REF1? might return the reference waveform data for reference channel 1.

REF<x>:DATE?

Returns the date that reference waveform data for channel <x> was copied into the internal reference memory, where x is the reference channel number. Query only.

Group

Vertical

Syntax

REF<x>:DATE?

Examples

REF1:DATE? might return the date the reference waveform data for reference channel 1 was created.

REF<x>:TIMe?

Returns the time that reference waveform data was copied into the internal reference memory for reference channel <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

REF<x>:TIMe?

Examples

REF2:TIMe? might return “16:54:05”.

REF<x>:HORizontal:DELay:TIMe?

Returns the horizontal delay time for reference waveform <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

REF<x>:HORizontal:DELay:TIMe?

Examples

`REF1:HORizontal:DELay:TIMe?` might return the horizontal delay time for reference waveform 1.

REF<x>:HORizontal:SCAle?

Returns the horizontal scale for reference waveform <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

`REF<x>:HORizontal:SCAle?`

Examples

`REF<x>:HORizontal:SCAle?` might return :REF1:HORIZONTAL:SCALE 4.0E-4.

REF<x>:POSition?

Returns the vertical position for channel <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

`REF<x>:POSition?`

Examples

`REF2:POSition?` might return the vertical position for reference 2.

REF<x>:VERTical:POSition?

Returns the vertical position of the reference waveform specified by <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

`REF<x>:VERTical:POSition?`

Examples

`REF1:VERTical:POSition?` might return :REF1:VERTICAL:POSITION -1.3000E+00 indicating that the current position of Reference 1 is 1.3 divisions below the center graticule.

REF<x>:VERTical:SCAle?

Returns the vertical scale for the reference waveform specified by <x>, where x is the reference channel number. Query only.

Group

Vertical

Syntax

`REF<x>:VERTical:SCAle?`

Examples

`REF2:VERTICAL:SCALE?` might return `:REF2:VERTICAL:SCALE 1.0000e+00` indicating that the current vertical scale setting for reference 2 is 1 V per division.

*RST

(Reset) Returns the instrument to a known set of instrument settings, but does not purge any stored settings. This command executes a subset of the FACtory command.

The *RST command does not change the following items:

- Calibration data that affects device specifications
- Output queue
- Service Request Enable Register settings
- Power-On Status Clear flag setting
- Alias definitions
- Stored settings or waveforms
- The *PUD? response
- Any of the values associated with the DATA command
- Instrument password

Group

Status and Error

Syntax

`*RST`

Related Commands

FACtory on page 73, **PSC* on page 129, **RCL* on page 130, *RECAll:SETUp* on page 130, **SAV* on page 135, *SAVe:SETUp* on page 137

Arguments

None

Examples

`*RST` resets the instrument settings to factory defaults.

S commands

This section lists commands and queries that begin with the letter S.

*SAV

Saves the state of the instrument into a specified nonvolatile memory location. You can later use the *RCL command to restore the instrument to this saved state. This is equivalent to selecting the Save Setup option in the Save/Recall Setup menu. Command only, no query form.

Group

Save and Recall

Syntax

`*SAV <NR1>`

Related Commands

FACtory on page 73, **RCL* on page 130, *RECAll:SETUp* on page 130

Arguments

`<NR1>` is an integer value in the range from 1 to 10 and specifies a memory location. Any settings that have been stored previously at this location are overwritten.

Examples

`*SAV 2` saves the current instrument settings in memory location 2.

SAVe:ASSIgn:TYPe

Sets or queries the assignment of the data to be saved when the front-panel Save button is pressed.

Group

Save and Recall

Syntax

SAVe:ASSIgn:TYPe {IMAGe|WAVEform|SETUp|}

SAVe:ASSIgn:TYPe?

Arguments

`IMAGe` assigns the Save button to save screen images.

`WAVEform` assigns the Save button to save waveforms.

`SETUp` assigns the Save button to save setups.

Examples

`SAVe:ASSIgn:TYPe WAVEform` sets the data to be saved to waveform.

SAVe:ASSIgn:TYPe? might return :SAVe:ASSIgn:TYPe WAVEform indicating that a wavefrom will be saved when the Save button is pressed.

SAVe:IMAge

Saves the screen image to a file. Command only, no query form.

Supported image formats are png, windows bitmap, and jpg. The format to use is determined by the value obtained from the :SAVe:IMAge:FILEFormat? query.

Group

Save and Recall

Syntax

SAVe:IMAge <file path>

Arguments

<file path> is a quoted string that defines the path and name of the screen image file to save. Use file name extensions that are appropriate for image format. If you do not specify a path to a folder, the instrument saves the screen image file in the current working folder, using the current save image file format. The current folder refers to the name of a folder as returned by the FILESystem:CWD query.

Use the SAVe:IMAge:FILEFormat command to set the screen image graphical file format.

Examples

SAVe:IMAge" usb0\PROD-TST\VID-EVAL.BMP" saves the screen image to the file VID-EVAL.BMP in the folder usb0\PROD-TST .

SAVe:IMAge:FILEFormat

Sets or queries the file format to use for saving screen images when the file type cannot be determined from the given file name or when screen images are captured by using the front panel.

Group

Save and Recall

Syntax

SAVe:IMAge:FILEFormat {PNG|BMP|JPG}

SAVe:IMAGe:FILEFormat?

Arguments

BMP sets the screen image file format to Microsoft Windows Bitmap format.

PNG saves the file in Portable Network Graphics format.

JPG saves the file in Joint Picture Group format.

Examples

`SAVe:IMAge:FILEFormat PNG` sets the screen image graphical file format to PNG.

SAVe:IMAGe:LAYout

Sets or returns the layout to use for saved screen images.

Group

Save and Recall

Syntax

`SAVe:IMAGe:LAYout {LANdscape|PORTRait}`

`SAVe:IMAGe:LAYout?`

Arguments

`LANdscape` specifies that screen images are saved in landscape format.

`PORTRait` specifies that screen images are saved in portrait format.

Examples

`SAVe:IMAGe:LAYout LANdscape` specifies that images are saved in landscape format.

`SAVe:IMAGe:LAYout?` might return `:SAVe:IMAGe:LAYout LANdscape` indicating that images are saved in landscape format.

SAVe:SETUp

Saves the current state of the instrument into the specified memory location. This is equivalent to selecting the Save Setup option in the Save/Recall Setup menu. You can later use the *RCL command to restore the instrument to this saved state. Command only, no query form.

Group

Save and Recall

Syntax

`SAVe:SETUp {<file path>|<NR1>}`

Related Commands

RECAll:SETUp on page 130, **RCL* on page 130, **SAV* on page 135

Arguments

`<NR1>` specifies a location for saving the current front-panel setup. The front-panel setup value ranges from 1 to 10. Using an out-of-range value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

`<file path>` is the target location for storing the setup file. <file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>:<dir>/<filename>. <extension> and one or more <dir>s are optional. If you do not specify them, the

instrument will store the file in the current working directory. <filename> stands for a filename. (Use of wildcard characters in filenames is not supported.) Filename extensions are not required but are highly recommended. For setups, use the extension .SET.

Examples

`SAVe:SETUp 5` saves the current front-panel setup to memory location 5.

`SAVE:SETUP "TEK00000.SET"` saves the current instrument setup in the file TEK00000.SET in the current working directory.

SAVe:WAVEform

Stores a waveform in one of the nonvolatile reference memory locations. This command is equivalent to selecting the Save Waveform option in the Save/Recall Waveform menu. Only individual analog waveforms (CH<x>, MATH, FFT) can be saved to reference memory locations. Command only, no query form.

You can save a specified waveform or waveforms to a single CSV file when the `SAVE:WAVEFORM:FILEFORMAT` is set to `SPREADSHEET`.

You can save a specified waveform or waveforms to consecutive ISF (internal save format) files when the `SAVE:WAVEFORM:FILEFORMAT` is set to `INTERNAL`.

Group

Save and Recall

Syntax

`SAVe:WAVEform[<wfm>,{REF<x>}] | [REF<x>, <QString>]`

Related commands

RECAll:WAVEForm on page 131, *SAVe:WAVEform:FILEFormat* on page 139

Arguments

`<wfm>`, `<REF<x>` saves the specified waveform to the specified reference memory location. `<wfm>` can be any live analog channel (where <x> is the channel number), the MATH waveform, or FFT waveform. `<wfm>`, `<QString>` saves the specified waveform to the file specified in the quoted string argument. Any live channel (such as CH1), the MATH waveform, the FFT waveform, or any reference waveform can be saved to a file.

`REF<x>` is one of the allowable reference waveform storage locations.

`<file path>` is a quoted string that defines the path and name of the waveform file to save. Use the extension .CSV for saved waveform files. Waveform data is saved as self-documented comma-separated ASCII values.

Examples

`SAVE:WAVEFORM CH1,REF1` saves the CH1 waveform in reference memory location 1.

`:SAVE:WAVEFORM:FILEFORMAT SPREADSHEET; :SAVE:WAVEFORM Ch1, "usb0/test1_ch1.csv"` saves channel 1 waveforms to usb0/test1_ch1.csv.

`:SAVe:WAVEform:FILEFormat INTERNal; :SAVe:WAVEform Ch1, "usb0/test1_ch1.isf"` saves channel 1 waveforms usb0/test1_ch1.isf

SAVe:WAVEform:FILEFormat

Specifies or queries the file format for saved waveforms. Waveform header and timing information is included in the resulting file of non-internal formats.

Group

Save and Recall

Syntax

`SAVe:WAVEform:FILEFormat {INTERNal|SPREADSheet}`

`SAVe:WAVEform:FILEFormat?`

Related commands

CURVe on page 55, *DATa* on page 57, *DATa:STARt* on page 58, *DATa:STOP* on page 59, *SAVe:WAVEform* on page 138, *WFMInpre:NR_Pt?* on page 166, *WFMInpre:NR_Pt?* on page 166

Arguments

`INTERNal` specifies that waveforms are saved in an internal format, using aisf filename extension. These files can be recalled as reference waveforms.

`SPREADSheet` specifies that waveform data is saved in a format that contains comma delimited values. These waveform data files are named using the csv filename extension. Saving waveforms in CSV format enables spreadsheet programs to import the data.

Examples

`SAVE:WAVEFORM:FILEFORMAT INTERNAL` specifies that the internal file format is the format used for saving waveforms.

`SAVE:WAVEFORM:FILEFORMAT?` might return `:SAVE:WAVEFORM:FILEFORMAT INTERNAL` indicating that waveforms are saved using the internal format.

SELect:CH<x>

Turns the display of the channel <x> waveform on or off, where <x > is the channel number. This command also resets the acquisition. The query returns whether the channel is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

`SELect:CH<x> {ON | OFF | <NR1>}`

`SELect:CH<x>?`

Arguments

`ON` turns on the display of the specified waveform. This waveform also becomes the selected waveform.

`OFF` turns off the display of the specified waveform.

`<NR1> = 0` turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:CH2 ON turns the channel 2 waveform display on, and selects channel 2.

SELECT:CH1? might return :SELECT:CH1 1 indicating that channel 1 is being displayed.

SELect:CONTROl

Sets or queries the waveform that is the recipient (focus) of future channel-related commands, for example, the cursor commands. The command form also performs the equivalent of a SELect:CH<x> ON command, as well as the Math, FFT and Reference of that command.

Group

Vertical

Syntax

SELect:CONTROl {CH<x> | MATH | FFT | REF<x>}

SELect:CONTROl?

Arguments

CH<x> specifies a channel waveform as the waveform affected by the front-panel controls. <x> is the channel number.

MATH specifies the math waveform as the waveform that is affected by the front-panel controls.

FFT specifies the FFT waveform as the waveform that is affected by the front-panel controls.

REF<x> specifies a reference waveform as the waveform affected by the front-panel controls. <x> specifies the reference waveform number (1 or 2).

Returns

NONE if all the channels are turned off. NONE is ignored on input.

Examples

SELect:CONTROl CH1 sets channel 1 as the recipient of future channel related commands.

SELect:CONTROl? might return :SELect:CONTROl CH1 indicating that channel 1 is the recipient of future channel related commands.

SELect:FFT

Turns on and off the display of the FFT waveform. The query returns whether the FFT waveform is on or off, but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

SELect:FFT {ON|OFF|<NR1>}

SELect:FFT?

Arguments

ON turns on the display of the specified waveform. This waveform also becomes the selected waveform.

OFF turns off the display of the specified waveform.

<NR1> = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:FFT ON turns the math waveform display on, and selects it.

SELECT:FFT? might return :SELECT:FFT 1 indicating that the math waveform is being displayed.

SELect:MATH

Turns on and off the display of the math waveform. The query returns whether the math waveform is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

SELect:MATH {ON|OFF|<NR1>}

SELect:MATH?

Arguments

ON turns on the display of the specified waveform. This waveform also becomes the selected waveform.

OFF turns off the display of the specified waveform.

<NR1> = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:MATH ON turns the math waveform display on, and selects it.

SELECT:MATH? might return :SELECT:MATH 1 indicating that the math waveform is being displayed.

SELect:REF<x>

Turns on and off the display of reference waveform <x>. The <x > variable represents the reference channel number. The query returns whether the channel is on or off.

Group

Vertical

Syntax

SELect:REF<x> {ON|OFF|<NR1>}

SELect:REF<x>?

Arguments

ON turns on the display of the specified waveform. This waveform also becomes the selected waveform.

OFF turns off the display of the specified waveform.

<NR1> = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:REF2 ON displays reference 2 and makes reference 2 the selected waveform.

SELECT:REF2? might return :SELECT:REF2 1 indicating that reference waveform 2 is being displayed.

SET?

Returns most instrument settings. You can send these responses back to the instrument to return the instrument to the state it was in when you sent SET. This query is identical to the *LRN? query. Query only.

Note: The SET? query always returns command headers, regardless of the setting of the HEADer command. This is because the returned data is intended to be able to be sent back to the instrument as concatenated commands. The VERBose command can still be used to specify whether the returned headers should be abbreviated or full length.

Group

Miscellaneous

Syntax

SET?

Related Commands

HEADer on page 87, **LRN?* on page 103

Returns

Most instrument settings. See *Appendix B: Factory Setup.*

Examples

SET? might return a partial string like the following: ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE; NUMAVG 16;:HEADER 1;:VERBOSE 1;:DISPLAY:FORMAT YT;STYLE VECTORS;PERSISTENCE 500.0E-3;CONTRAST 50;:LOCK NONE;:HARDCOPY:FORMAT EPSON;PORT RS232;LAYOUT PORTRAIT;

SETUP<x>:DATE?

Returns the date when the instrument setup was saved for the specified setup<x>. Query only.

Group

Save and Recall

Syntax

SETUP<x>:DATE?

Examples

SETUP4:DATE? might return :SETUP4:DATE: 04-18-06 which is the setup date for channel 4.

SETUP<x>:TIME? (Query Only)

Returns the time when the instrument setup was saved for the specified setup<x>.

Group

Save and Recall

Syntax

SETUP<x>:TIME?

Examples

SETUP2:TIME? might return :SETUP2:TIME: 15:24:07 which is the setup time for channel 2. The default port is 4000.

*SRE

(Service Request Enable) sets or queries the bits in the Service Request Enable Register (SRER). Refer to the Status and Events section for more information.

Group

Status and Error

Syntax

*SRE <NR1>

*SRE?

Related Commands

*CLS on page 46, DESE on page 61, *ESE on page 69, *ESR? on page 70, EVENT? on page 70, EVMsg? on page 71, FACtory on page 73, *PSC on page 129

Arguments

<NR1> is an integer value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error. The power-on default for SRER is 0 if *PSC is 1. If *PSC is 0, the SRER maintains its value through a power cycle.

Examples

*SRE 48 sets the bits in the SRER to 00110000 binary.

*SRE? might return a value of 32, showing that the bits in the SRER have the binary value 00100000.

*STB?

(Read Status Byte) query returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. Refer to the Status and Events section for more information. Query only.

Group

Status and Error

Syntax

*STB?

Related Commands

CLS on page 46, *DESE* on page 61, *ESE* on page 69, *ESR?* on page 70, *EVENT?* on page 70, *EVMsg?* on page 71, *FACtory* on page 73, *SRE* on page 143

Returns

<NR1> is the contents of the Status Byte Register (SBR)

Examples

*STB? might return the value 96, showing that the SBR contains the binary value 01100000.

T commands

This section lists commands and queries that begin with the letter T.

TEKSecure

Sets the Teksecure Erase Memory option to erase user data, which may be settings or confidential data files. This is equivalent to invoking Teksecure from the Utility->Config->TekSecure Erase Memory menu. This is a time-consuming operation (3 to 5 minutes) and the instrument is inoperable until the TekSecure operation is complete.

Group

Miscellaneous

Syntax

TEKSecure

TIMe

Sets or queries the time the instrument displays. The instrument uses the time and date values to time stamp files and show the time and date on the instrument display.

Conditions

Group

Miscellaneous

Syntax

`TIMe <QString>`

`TIMe?`

Related commands

DATE on page 60

Arguments

`<QString>` is a time in the form "hh:mm:ss" where hh refers to a two-digit hour number, mm refers to a two-digit minute number from 00 to 59, and ss refers to a two-digit second number from 00 to 59.

Examples

`TIME "14:00:00"` sets the time to exactly 2:00 p.m.

`TIME?` might return `:TIME "11:25:03"` indicating that the current time is set to 11:25 a.m. and 03 seconds.

TRIGger

Forces a trigger event to occur. No query form.

Group

Trigger

Syntax

```
TRIGger FORCe
```

Arguments

`FORCe` creates a trigger event. If TRIGger:STATE is READy, the acquisition will complete; otherwise this command is ignored.

Examples

`TRIGger FORCE` forces a trigger event to occur.

TRIGger:A

Sets the instrument trigger level to 50% of the minimum and maximum values of the signal. Returns the current A trigger settings when used as a query.

The trigger level is the voltage threshold through which the trigger source signal must pass to generate a trigger event. This command works for the following cases: Edge Trigger (when source is not Line), Logic Trigger (when Clock Source is not Off or Logic Pattern is Don't Care), and Pulse Width Trigger.

Group

Trigger

Syntax

```
TRIGger:A SETLevel
TRIGger:A?
```

Related commands

TRIGger:A:EDGE? on page 147, *TRIGger:A:PULse?* on page 152

Arguments

`SETLevel` sets the A trigger level to 50% of the minimum and maximum values of the trigger source input signal. This is equivalent to pressing the front-panel PUSH to SET to 50% button.

Examples

`TRIGger:A SETLEVEL` sets the A trigger level to 50% of the range of the minimum and maximum values of the trigger input signal.

`TRIGGER:A?` might return a long response with A trigger parameters, some of which could be as follows: `:TRIGGER:A:MODE AUTO;TYPE EDGE;LEVEL 20.0000E-3;LEVEL:CH1 20.0000E-3;CH2 0.0000; :TRIGGER:A:UPPERTHRESHOLD:CH1 1.4000;CH2 800.0000E-3;:TRIGGER:A:LOWERTHRESHOLD:CH1 20.0000E-3;CH2`

```
0.0000;:TRIGGER:A:HOLDOFF:TIME 20.0000E-9;:TRIGGER:A:EDGE:SOURCE CH1;COUPLING
DC;SLOPE RISE;:TRIGGER:A:LOGIC:CLASS SETHOLD;FUNCTION AND;THRESHOLD: CH1
20.0000E-3;CH2 0.0000; :TRIGGER:A:LOGIC:INPUT:CH1 X;CH2 X; CLOCK:SOURCE NONE;EDGE.
```

TRIGger:A:EDGE?

Returns the trigger coupling, source, and slope settings for the A edge trigger. Query only.

Group

Trigger

Syntax

`TRIGger:A:EDGE?`

Related commands

TRIGger:A:PULse? on page 152

Returns

Trigger coupling, source, and slope settings for the A edge trigger.

Examples

`TRIGger:A:EDGE?` might return `:TRIGGER:A:EDGE:SOURCE CH1;COUPLING DC; SLOPE RISE` indicating the trigger source, coupling, and slope for the A edge trigger.

TRIGger:A:EDGE:COUPling

Sets or queries the type of coupling for the A edge trigger. This is equivalent to setting the Coupling option in the Trigger menu.

Group

Trigger

Syntax

`TRIGger:A:EDGE:COUPling {DC|HFRej|LFRej|NOISErej}`

`TRIGger:A:EDGE:COUPling?`

Related commands

TRIGger:A:EDGE:SOUrce on page 148, *TRIGger:A:EDGE:SLOpe* on page 148

Arguments

`DC` selects DC coupling, which passes all input signals to the trigger circuitry.

`HFRej` coupling attenuates the high-frequency components, above 50 kHz, of the trigger signal.

`LFRej` coupling removes the low-frequency components, below 50 kHz, of the trigger signal.

NOISErej selects DC low sensitivity, which provides stable triggering by increasing the trigger hysteresis. Increased hysteresis reduces the trigger sensitivity to noise but may require greater trigger signal amplitude.

Examples

TRIGger:A:EDGE:COUPling DC sets the A edge trigger coupling to DC.

TRIGGER:A:EDGE:COUPLING? might return :TRIGGER:A:EDGE:COUPLING DC indicating that the A edge trigger coupling is set to DC.

TRIGger:A:EDGE:SLOpe

Sets or queries the slope for the A edge trigger. This is equivalent to setting the Slope option in the Trigger menu.

Group

Trigger

Syntax

TRIGger:A:EDGE:SLOpe{RISe|FALL}

TRIGger:A:EDGE:SLOpe?

Related commands

TRIGger:A:EDGE:SOUrce on page 148, *TRIGger:A:EDGE:COUPling* on page 147

Arguments

FALL specifies to trigger on the falling or negative edge of a signal.

RISe specifies to trigger on the rising or positive edge of a signal.

Examples

TRIGger:A:EDGE:SLOpe RISE sets the A edge trigger to occur on the rising edge of the signal.

TRIGGER:A:EDGE:SLOPE? might return :TRIGGER:A:EDGE:SLOPE FALL indicating that the A edge trigger slope is negative.

TRIGger:A:EDGE:SOUrce

Sets or queries the source for the edge trigger. This is equivalent to setting the Source option in the Trigger menu.

Group

Trigger

Syntax

TRIGger:A:EDGE:SOUrce {CH1 | CH2 | LINE | AUX}

TRIGger:A:EDGE:SOUrce?

Related commands

TRIGger:A:EDGE:SLOpe on page 148, *TRIGger:A:EDGE:COUPling* on page 147

Arguments

CH<x> specifies one of the analog input channels. The value of <x> can vary from 1 through 4 for 4-channel instruments or 1 through 2 for 2-channel instruments.

AC LINE specifies the AC line as a trigger source.

AUX specifies the Aux IN as a trigger source.

Examples

TRIGger:A:EDGE:SOUrce CH1 specifies channel 1 as the A edge trigger source.

TRIGger:A:EDGE:SOUrce? might return :TRIGGER:A:EDGE:SOURCE CH1 indicating that channel 1 is the A edge trigger source.

TRIGger:A:HOLDOff?

Returns the A trigger holdoff parameters. These parameters specify the time period during which the trigger circuitry is not looking to generate a trigger event. Query only.

Group

Trigger

Syntax

TRIGger:A:HOLDOff?

Related commands

TRIGger:A:HOLDOff:TIMe on page 149

Returns

A trigger holdoff value.

Examples

TRIGger:A:HOLDOff? might return :TRIGGER:A:HOLDOFF:TIME 900.0000E-09, indicating that the A edge trigger holdoff time (by default) is 900 ns.

TRIGger:A:HOLDOff:TIMe

Sets or queries the A trigger holdoff time.

Group

Trigger

Syntax

TRIGger:A:HOLDOff:TIMe <NR3>

TRIGger:A:HOLDOff:TIMe?

Arguments

<NR3> is the A trigger holdoff time. The range is 20 ns to 8.0 s.

Examples

TRIGger:A:HOLDOff:TIMe 10 sets the holdoff time to 10 s.

TRIGGER:A:HOLDOFF:TIME? might return :TRIGGER:A:HOLDOFF:TIME 1.2000E-06 indicating that the A trigger holdoff time is set to 1.2 µs.

TRIGger:A:LEVel

Sets or queries the trigger level for the A trigger. This command is equivalent to adjusting the front-panel TRIGGER LEVEL knob.

Note: When the edge trigger source is set to AC LINE, the instrument ignores the set form of the command.

When the edge trigger source is set to AC LINE, the query form of the command returns zero.

Group

Trigger

Syntax

TRIGger:A:LEVel{ECL|TTL|<NR3>}

TRIGger:A:LEVel?

Arguments

<NR3> specifies the trigger level in user units (usually volts).

ECL specifies a preset ECL high level of -1.3V.

TTL specifies a preset TTL high level of 1.4V.

Examples

TRIGGER:A:LEVEL TTL sets the A edge trigger to TTL high level, which is 1.4 V.

TRIGger:A:LEVel? might return :TRIGGER:A:LEVel 1.3000E+00 indicating that the A edge trigger is set to 1.3 V.

TRIGger:A:LEVel:CH<x>

Sets or queries the trigger level for the specified channel. Each channel can have an independent level.

Group

Trigger

Syntax

TRIGger:A:LEVel:CH<x> {<NR3>|TTL|ECL}

TRIGger:A:LEVel:CH<x>?

Arguments

ECL specifies a preset ECL high level of -1.3V.

TTL specifies a preset TTL high level of 1.4V.

<NR3> specifies the trigger level in user units (usually volts).

Examples

TRIGGER:A:LEVEL: CH1 TTL sets the A edge trigger to TTL high level for channel 1.

TRIGGER:A:LEVEL:CH2? might return :TRIGGER:A:LEVEL:CH2 1.3000E+00 indicating that the A edge trigger is set to 1.3 V for channel 2.

TRIGger:A:LOWerthreshold:CH<x>

Sets or queries the lower threshold for the channel selected. Each channel can have an independent level. Used in Runt trigger as the lower threshold. Used for all other trigger types as the single level/threshold.

Group

Trigger

Syntax

TRIGger:A:LOWerthreshold:CH<x> {ECL|TTL|<NR3>}

TRIGger:A:LOWerthreshold:CH<x>?

Related commands

TRIGger:A:LEVel:CH<x> on page 150

Arguments

ECL specifies a preset ECL high level of -1.3 V.

TTL specifies a preset TTL high level of 1.4 V.

<NR3> is the clock level, in volts.

Examples

TRIGGER:A:LOWERTHRESHOLD:CH2 50E-3 sets the lower limit threshold for CH2 of the pulse runt trigger to 50 mV.

TRIGGER:A:LOWERTHRESHOLD:CH2? might return :TRIGGER:A: LOWERTHRESHOLD:CH2 1.2000E-01 indicating that the lower limit threshold for CH2 of the pulse runt trigger is set to 120 mV.

TRIGger:A:MODe

Sets or queries the trigger mode.

Group

Trigger

Syntax

`TRIGger:A:MODe {AUTO|NORMal}`

`TRIGger:A:MODe?`

Related Commands

TRIGger:A:LEVel on page 150

Arguments

`AUTO` generates a trigger if a trigger is not detected within a specific time period.

`NORMal` waits for a valid trigger event.

Examples

`TRIGger:A:MODe NORMAL` specifies that a valid trigger event must occur before a trigger is generated.

`TRIGGER:A:MODE ?` might return `:TRIGGER:A:MODE NORMAL` indicating that a valid trigger event must occur before a trigger is generated.

TRIGger:A:PULse?

Returns the current Pulse Trigger settings. Query only.

Group

Trigger

Syntax

`TRIGger:A:PULse?`

Related commands

TRIGger:A:EDGE? on page 147

Examples

`TRIGger:A:PULse?` might return `:TRIGGER:A:PULSE:CLASS TRAnsITION`.

TRIGger:A:PULse:CLAss

Sets or queries the type of pulse on which to trigger.

Group

Trigger

Syntax

`TRIGger:A:PULse:CLAss {RUNt|WIDth}`

`TRIGger:A:PULse:CLAss?`

Related commands

TRIGger:A:RUNT? on page 156, *TRIGger:A:PULSE:Width?* on page 153, *TRIGger:A:TYPe* on page 158

Arguments

`RUNt` triggers when a pulse crosses the first preset voltage threshold but does not cross the second preset threshold before recrossing the first.

`WIDth` triggers when a pulse is found that has the specified polarity and is either inside or outside the specified time limits.

Examples

`TRIGGER:A:PULSE:CLASS WIDTH` specifies a width pulse for the A trigger.

`TRIGGER:A:PULSE:CLASS?` might return `:TRIGGER:A:PULSE:CLASS WIDTH` indicating that a pulse was found that is of the specified polarity and width.

TRIGger:A:PULSE:Width?

Queries the width for the pulse-width trigger. Query only.

Group

Trigger

Syntax

`TRIGger:A:PULSEWidth?`

Examples

`TRIGger:A:PULSEWidth?` might return `:TRIGGER:A:PULSEWIDTH:POLARITY POSITIVE;WHEN LESSTHAN;WIDTH 8.0E-9`

TRIGger:A:PULse:WIDth:POLarity

Sets or queries the polarity for the pulse width trigger. This is equivalent to setting the Polarity option in the Pulse Trigger menu.

Group

Trigger

Syntax

`TRIGger:A:PULse:WIDth:POLarity {NEGative|POSitive}`

`TRIGger:A:PULse:WIDth:POLarity?`

Arguments

`POSITIVe` polarity specifics pulses with a rising leading edge.

`NEGAtive` polarity specifies pulses with a falling leading edge.

Examples

`TRIGGER:A:PULSEWIDTH:POLARITY NEGATIVE` sets the pulse polarity to negative.

`TRIGGER:A:PULSEWIDTH:POLARITY?` might return `:TRIGGER:A:WIDTH:POLARITY POSITIVE` indicating a positive pulse.

TRIGger:A:PULSEWidth:SOUrce

Sets or queries the source for the pulse width trigger. This is equivalent to setting the Source option in the Trigger menu.

Group

Trigger

Syntax

`TRIGger:A:PULse:SOUrce {CH1|CH2| LINE}`

`TRIGger:A:PULse:SOUrce?`

Arguments

`CH<x>` specifies one of the analog input channels. The value of <x> can be 1 through 4 on four channel instruments, or 1 or 2 on two channel instruments.

`LINE` specifies AC line voltage.

Examples

`TRIGGER:A:PULSEWIDTH:SOURCE CH1` sets channel 1 as the pulse width source.

`TRIGGER:A:PULSEWIDTH:SOURCE?` might return `:TRIGGER:A:PULSEWIDTH:SOURCE CH1` indicating that channel 1 is the pulse width source.

TRIGger:A:PULse:WIDth:WHEN

Sets or queries whether to trigger on a pulse that meets, falls outside, or within the specified range of limits. This is equivalent to setting the When option in the Pulse Trigger menu.

Group

Trigger

Syntax

`TRIGger:A:PULse:WIDth:WHEN {LESSthan|MOREthan|EQual|UNEQual}`

`TRIGger:A:PULse:WIDth:WHEN?`

Related commands

TRIGger:A:PULse:WIDth:WIDth on page 155

Arguments

`LESSthan` sets the instrument to trigger if a pulse is detected with width less than the time set by the `TRIGger:A:PULSEWidth:WIDth` command.

`MOREthan` sets the instrument to trigger if a pulse is detected with width more than the time set by the `TRIGger:A:PULSEWidth:WIDth` command.

`EQUal` sets the instrument to trigger if a pulse is detected with width equal to the time period specified in `TRIGger:A:PULSEWidth:WIDth` within a ±5% tolerance.

`UNEQual` sets the instrument to trigger if a pulse is detected with width greater than or less than (but not equal) the time period specified in `TRIGger:A:PULSEWidth:WIDth` within a ±5% tolerance.

Examples

`TRIGGER:A:PULSEWIDTH:WHEN LESSTHAN` specifies that the duration of the A pulse will fall within defined high and low limits.

`TRIGGER:A:PULSEWIDTH:WHEN?` might return `:TRIGGER:A:PULSEWIDTH:WHEN LESSTHAN` indicating that the conditions for generating a width trigger.

TRIGger:A:PULse:WIDth:WIDth

Sets or queries the width setting for the pulse width trigger. This is equivalent to setting the Width option by using the Pulse Trigger menu and the TRIGGER knob.

Group

Trigger

Syntax

`TRIGger:A:PULse:WIDth:WIDth <NR3>`

`TRIGger:A:PULse:WIDth:WIDth?`

Related commands

TRIGger:A:PULse:WIDth:WHEN on page 155

Arguments

<NR3> specifies the pulse width, in seconds.

Examples

TRIGGER:A:PULSEWIDTH:WIDTH 5.0E-6 sets the pulse width to 5 µs.

TRIGGER:A:PULSEWIDTH:WIDTH? might return :TRIGGER:A:PULSEWIDTH:WIDTH 2.0000E-9 indicating that the pulse width is set to 2 ns.

TRIGger:A:RUNT?

Returns the current A runt trigger parameters. Query only.

Group

Trigger

Syntax

TRIGger:A:RUNT?

Examples

TRIGGER:A:RUNT? might return :TRIGGER:A:RUNT:SOURCE CH1;POLARITY POSITIVE;WHEN OCCURS;WIDTH 4.0000E-9.

TRIGger:A:RUNT:POLarity

Sets or queries the polarity for the runt trigger.

Group

Trigger

Syntax

TRIGger:A:RUNT:POLarity {NEGative|POSitive}

TRIGger:A:RUNT:POLarity?

Arguments

POSitive indicates that the rising edge crosses the low threshold and the falling edge recrosses the low threshold without either edge ever crossing the high threshold.

NEGative indicates that the falling edge crosses the high threshold and the rising edge recrosses the high threshold without either edge ever crossing the low threshold.

Examples

TRIGGER:A:RUNT:POLARITY NEGATIVE specifies that the polarity of the A pulse runt trigger is negative.

TRIGGER:A:RUNT:POLARITY? might return :TRIGGER:A:RUNT:POLARITY POSITIVE indicating that the polarity of the A pulse runt trigger is positive.

TRIGger:A:RUNT:SOUrce

Sets or queries the source for the A runt trigger.

Group

Trigger

Syntax

TRIGger:A:RUNT:SOUrce {CH1|CH2}

Arguments

CH1–CH2 specifies an analog input channel as the trigger source.

Examples

TRIGGER:A:RUNT:SOURCE CH1 sets channel 2 as the source for the A pulse trigger.

TRIGGER:A:RUNT:SOURCE? might return :TRIGGER:A:RUNT:SOURCE CH2 indicating that channel 2 is the source for the A pulse trigger.

TRIGger:A:RUNT:WHEn

Sets or queries the type of pulse width the trigger checks for when it detects a runt.

Group

Trigger

Syntax

TRIGger:A:RUNT:WHEn {LESSthan|MOREthan|EQual|UNEQual|OCCURS}

TRIGger:A:RUNT:WHEn?

Related commands

TRIGger:A:RUNT:WIDth on page 158

Arguments

OCCURS specifies a trigger event if a runt of any detectable width occurs.

LESSthan sets the instrument to trigger if a runt pulse is detected with a width less than the time set by the TRIGger:A:RUNT:WIDth command.

MOREthan sets the instrument to trigger if a runt pulse is detected with a width more than the time set by the TRIGger:A:RUNT:WIDth command.

EQUal sets the instrument to trigger if a runt pulse is detected with a width equal to the time period specified in TRIGger:A:RUNT:WIDth within a ±5% tolerance.

UNEQual sets the instrument to trigger if a runt pulse is detected with a width greater than or less than (but not equal to) the time period specified in TRIGger:A:RUNT:WIDth within a ±5% tolerance.

Examples

TRIGGER:A:RUNT:WHEN THAN sets the runt trigger to occur when the instrument detects a runt in a pulse wider than the specified width.

TRIGGER:A:RUNT:WHEN? might return :TRIGGER:A:PULSE:RUNT:WHEN OCCURS indicating that a runt trigger will occur if the instrument detects a runt of any detectable width.

TRIGger:A:RUNT:WIDth

Sets or queries the width for a runt trigger.

Group

Trigger

Syntax

TRIGger:A:RUNT:WIDth <NR3>

TRIGger:A:RUNT:WIDth?

Related commands

TRIGger:A:RUNT:WHEn on page 157

Arguments

<NR3> specifies the minimum width, in seconds.

Examples

TRIGGER:A:RUNT:WIDTH 15E-6 sets the minimum pulse width of the runt trigger to 15 µs.

TRIGGER:A:RUNT:WIDTH? might return :TRIGGER:A:PULSE:RUNT:WIDTH 2.0000E-09 indicating that the minimum pulse width of a runt trigger is 2 ns.

TRIGger:A:TYPe

Sets or queries the type of A trigger. This is equivalent to setting the Type option in the Trigger menu.

Group

Trigger

Syntax

TRIGger:A:TYPe{EDGe|PULSe}

TRIGger:A:TYPe?

Related commands

TRIGger:A:EDGE? on page 147, *TRIGger:A:PULse:CLAss* on page 153

Arguments

EDGE is a normal trigger. A trigger event occurs when a signal passes through a specified voltage level in the specified direction and is controlled by the TRIGger:A:EDGE commands.

PULse specifies that a trigger occurs when the specified signal meets the pulse width criteria that is controlled by the TRIGger:A:PULse commands.

Examples

TRIGGER:A:TYPE EDGE sets the A trigger type to EDGE.

TRIGGER:A:TYPE? might return :TRIGGER:A:TYPE PULSE indicating that the A trigger type is a pulse trigger.

TRIGger:A:UPPerthreshold:CH<x>

Sets or queries the upper threshold for channel <x>, where x is the channel number. Each channel can have an independent level. Used only for runt trigger type.

Group

Trigger

Syntax

TRIGger:A:UPPerthreshold:CH<x> {<NR3>|ECL|TTL}

TRIGger:A:UPPerthreshold:CH<x>?

Arguments

<NR3> is the threshold level in volts.

ECL specifies a preset ECL high level of -1.3 V.

TTL specifies a preset TTL high level of 1.4 V.

Examples

TRIGGER:A:UPPERTHRESHOLD:CH2 50E-3 sets the upper limit of the pulse runt trigger to 50 mV for channel 2.

TRIGGER:A:UPPERTHRESHOLD:CH2? might return :TRIGGER:A:UPPERTHRESHOLD:CH2 1.2000E-01 indicating that the upper limit of the pulse runt trigger is set to 120 mV.

TRIGger:FREQuency?

Returns the edge or pulse width trigger frequency. This is the same as the readout in the lower right corner of the screen. Query only.

Group

Trigger

Syntax

`TRIGger:FREQuency?`

Returns

Edge or pulse width trigger frequency.

Examples

`TRIGger:FREQuency?` might return `TRIGGER:FREQUENCY 1.0E3.`

TRIGger:STATE?

Returns the current state of the triggering system. Query only.

Group

Trigger

Syntax

`TRIGger:STATE?`

Related commands

TRIGger:A:MODe on page 152

Returns

`ARMED` indicates that the instrument is acquiring pretrigger information. All triggers are ignored when TRIGger:STATE is ARMed.

`AUTO` indicates that the instrument is in the automatic mode and acquires data even in the absence of a trigger.

`READY` indicates that all pretrigger information has been acquired and that the instrument is ready to accept a trigger.

`SAVE` indicates that the instrument is in save mode and is not acquiring data.

`TRIGGER` indicates that the instrument triggered and is acquiring the post trigger information.

Examples

`TRIGGER:STATE?` might return `:TRIGGER:STATE ARMED` indicating that the pretrigger data is being acquired.

U commands

This section lists commands and queries that begin with the letter U.

UNLock

Unlocks the front panel. This command is equivalent to LOCk NONe. Command only, no query form.

Group

Miscellaneous

Syntax

`UNLock ALL`

Related commands

LOCk on page 102

Arguments

`ALL` specifies all front-panel buttons.

Examples

`UNLock ALL` unlocks all front-panel buttons and knobs so they can be used.

V commands

This section lists commands and queries that begin with the letter V.

VERBose

Sets and queries the Verbose state that controls the length of keywords on query responses. Keywords can be both headers and arguments. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk).

Group

Miscellaneous

Syntax

`VERBose`

`VERBose?`

Related Commands

HEADer on page 87, **LRN?* on page 103

Arguments

`ON` or `<NR1>` ≠ 0 sets the Verbose state true, which returns full-length keywords for applicable setting queries.

`OFF` or `<NR1>` = 0 sets the Verbose state false, which returns minimum-length keywords for applicable setting queries.

Examples

`VERBose ON` sets the Verbose state true.

`VERBose?` might return the value `1`, showing that the Verbose state is true.

W commands

This section lists commands and queries that begin with the letter W.

*WAI

Prevents the instrument from executing further commands or queries until all pending commands that generate an OPC message are complete. This command allows you to synchronize the operation of the instrument with your application program. Command only, no query form.

The *WAI command will stop execution until certain instrument operations are complete. See *Table 11* on page 188.

Group

Status and Error

Syntax

```
*WAI
```

Related Commands

BUSY? on page 32, **OPC* on page 128

Examples

`*WAI` prevents the instrument from executing any further commands or queries until all pending commands that generate an OPC message are complete.

WAVFrm?

Returns WFMOutpre? and CURVe? data for the waveform as specified by the DATA:SOUrce command. This command is equivalent to sending both WFMOutpre? and CURVe?, with the additional provision that the response to WAVFrm? is guaranteed to provide a synchronized preamble and curve. The source waveform, as specified by :DATA:SOURCE, must be active or the query will not return any data and will generate an error indicator. Query only.

Group

Waveform

Syntax

```
WAVFrm?
```

Related Commands

CURVe on page 55, *DATa:SOUrce* on page 58, *WFMOutpre?* on page 170

Returns

See WFMPre? and CURVe? commands.

WFMInpre?

Returns the waveform formatting and scaling specifications to be applied to the next incoming CURVe command data. Query only.

Group

Waveform

Syntax

`WFMInpre?`

Related commands

CURVe on page 55, *DATa:SOUrce* on page 58, *WFMOutpre?* on page 170

Returns

Returns the response in the following format:

WFMPre:<wfm>:WFID <Qstring>;PT_FMT { ENV | Y }; XINcr <NR3>;PT_Off <NR1>;XZEro <NR3>;XUNit <QString>; YMUlt <NR3>;YZEro <NR3>;YOFF <NR3>;YUNit <QString>; NR_Pt <NR1>

Examples

`WFMINPRE?` might return the waveform formatting as `:WFMINPRE:BIT_NR8;BN_FMT RI;BYT_NR 1; BYT_OR MSB;ENCDG BIN;NR_PT 500;PT_FMTY; PT_OFF 0;XINCR 2.0000E-6;XZERO 1.7536E-6; XUNIT "s";YMULT 1.0000E-3;YOFF 0.0000; YZERO 0.0000;YUNIT "V"`.

WFMInpre:BIT_Nr

Sets or returns the number of bits per binary waveform point for the incoming waveform. Changing the value of WFMInpre:BIT_Nr also changes the value of WFMInpre:BYT_Nr.

Group

Waveform

Syntax

`WFMInpre:BIT_Nr`

`WFMInpre:BIT_Nr?`

Arguments

`<NR1>` is either 8 or 16.

Examples

`WFMINPRE:BIT_NR 16` sets the number of bits per waveform point to 16, for incoming data.

`WFMINPRE:BIT_NR?` might return `:WFMINPRE:BIT_NR 8` indicating that incoming waveform data uses 8 bits per waveform point.

WFMInpre:BYT_Nr

Sets or queries the data width for the incoming waveform. Changing the value of WFMInpre:BYT_Nr also changes the value of WFMInpre:BIT_Nr.

Group

Waveform

Syntax

`WFMInpre:BYT_Nr`

`WFMInpre:BYT_Nr?`

Arguments

`<NR1>` is an integer in the range of 1 to 2 that sets the number of bytes per point.

Examples

`WFMINPRE:BYT_NR 1` sets the number of bytes per incoming waveform data point to 1, which is the default setting.

`WFMINPRE:BYT_NR?` might return `:WFMINPRE:BYT_NR 2` indicating that there are 2 bytes per incoming waveform data point.

WFMInpre:ENCdg

Sets or queries the type of encoding for waveform data transferred with the CURVe command.

Group

Waveform

Syntax

`WFMInpre:ENCdg {ASCii|BINary}`

`WFMInpre:ENCdg?`

Arguments

`ASCii` specifies ASCII curve data.

`BINary` specifies that the incoming data is in a binary format whose further interpretation requires knowledge of BYT_NR, BIT_NR, BN_FMT, and BYT_OR.

Examples

`WFMINPre:ENCdg ASC` specifies that the waveform data is in ASCII format.

`WFMINPre:ENCdg?` might return `:WFMINPRE:ENCDG BIN`, indicating that the waveform data is in binary format.

WFMInpre:NR_Pt?

Returns the number of points that are in the incoming waveform record.

Group

Waveform

Syntax

`WFMInpre:NR_Pt <NR1>`

`WFMInpre:NR_Pt?`

Related Commands

CURVe on page 55, *DATa* on page 57, *DATa:STARt* on page 58, *DATa:STOP* on page 59, *SAVe:WAVEform* on page 138, *SAVe:WAVEform:FILEFormat* on page 139, *WFMInpre:NR_Pt?* on page 166

Arguments

`<NR1>` is the number of data points if `WFMInpre:PT_Fmt` is set to Y. It is the number of min-max pairs if `WFMInpre:PT_Fmt` is set to ENV.

Examples

`WFMINPRE:NR_PT 10000` specifies that 10000 data points will be expected.

`WFMINPRE:NR_PT ?` might return `:WFMINPRE:NR_PT 2000` indicating that there are 2000 data points in the expected incoming waveform record.

WFMInpre:XINcr

Sets or queries the horizontal interval between incoming waveform points in units specified by `WFMInpre:XUNit`.

Group

Waveform

Syntax

`WFMInpre:XINcr <NR3>`

`WFMInpre:XINcr?`

Arguments

`<NR3>` is the interval between points in the waveform record, in the units specified by WFMPre:XUNit.

Examples

`WFMINPRE:XINCR 3E-3` sets the interval between incoming waveform points to 3 ms.

`WFMINPRE:XINCR ?` might return `:WFMINPRE:XINCR 1.0000E-3` indicating that if `WFMInpre:XUNit` is set to "s", there is a 1 ms interval between incoming waveform points.

WFMInpre:XUNit

Sets or queries the horizontal units of the incoming waveform.

Group

Waveform

Syntax

`WFMInpre:XUNit <Qstring>`

`WFMInpre:XUNit?`

Related commands

WFMOutpre:XUNit? on page 175

Arguments

`<Qstring>` contains a maximum of three alpha characters that represent the horizontal unit of measure for the incoming waveform.

Examples

`WFMINPRE:XUNIT "HZ"` specifies that the horizontal units for the incoming waveform are hertz.

`WFMINPRE:XUNIT?` might return `:WFMINPRE:XUNIT "s"` indicating that the horizontal units for the incoming waveform are seconds.

WFMInpre:XZEro

Sets or queries the position value, in XUNits, of the first sample of the incoming waveform, relative to the trigger.

The instrument sets WFMPre:XZEro to zero when:

- The display mode is set to XY.
- The DATa:SOUrce is set to MATH FFT when the waveform is acquired.

Note: The instrument uses XZEro when calculating cursor readouts.

Group

Waveform

Syntax

`WFMInpre:XZEro <NR3>`

`WFMInpre:XZEro?`

Related commands

WFMInpre:XINcr on page 166, *WFMInpre:BYT_Nr* on page 165, *WFMOutpre:XZEro?* on page 175

Arguments

<NR3> argument is the floating point value of the position, in XUNits, of the first sample in the incoming waveform. If XUNits is "s", <NR3> is the time of the first sample in the incoming waveform.

Examples

WFMINPRE:XZERO 5.7E-6 indicates the trigger occurred 5.7 µs before the first sample in the waveform.

WFMINPRE:XZERO? might return :WFMINPRE:XZEro -7.5000E-6 indicating that the trigger occurs 7.5 µs after the first sample in the waveform.

WFMInpre:YMUlt

Sets or queries the vertical scale factor of the incoming waveform, expressed in YUNits per waveform data point level. For one byte waveform data, there are 256 data point levels. For two byte waveform data there are 65,536 data point levels. YMUlt, YOFf, and YZEro are used to convert waveform record values to YUNit values using the following formula (where dl is the data level; curve_in_dl is a data point in CURVe): value_in_units = ((curve_in_dl - YOFf_in_dl) * YMUlt) + YZEro_in_units.

Group

Waveform

Syntax

WFMInpre:YMUlt <NR3>

WFMInpre:YMUlt?

Related commands

DATa:DESTination on page 57, *WFMInpre:BYT_Nr* on page 165, *WFMInpre:YUNit* on page 169

Arguments

<NR3> is the vertical scale factor per digitizing level of the incoming waveform points.

Examples

WFMINPRE:YMULT? might return :WFMINPRE:YMULT 40.0000E-3, which (if YUNit is "V") indicates that the vertical scale is 40 mV/digitizing level (1V/div for 8-bit data).

WFMInpre:YOFf

Sets or queries the vertical position of the incoming waveform in digitizing levels. Variations in this number are analogous to changing the vertical position of the waveform.

Group

Waveform

Syntax

WFMInpre:YOFf <NR3>

WFMInpre:YOFf?

Arguments

<NR3> is a value expressed in digitizing levels.

Examples

WFMINPRE:YOFF 50 specifies that the zero reference point for the incoming waveform is 50 digitizing levels (2 divisions, for 8-bit data) above the center of the data range.

WFMINPRE:YOFF? might return :WFMINPRE:YOFF 25 indicating the vertical position of the incoming waveform in digitizing levels.

WFMInpre:YUNit

Sets or returns the vertical units of the incoming waveform.

Group

Waveform

Syntax

WFMInpre:YUNit <Qstring>

WFMInpre:YUNit?

Arguments

<Qstring> contains a maximum of three alpha characters that represent the vertical unit of measure for the incoming waveform.

Returns

The query may return the following:

- Volts for volts
- U for unknown units (divisions)
- dB for decibels
- ? for unknown mask waveforms units
- A for amperes
- VA for volt × amperes
- AA for amperes × amperes
- VV for volts × volts

Examples

WFMINPRE:YUNIT "A" specifies that the vertical units for the incoming waveform are Amperes.

WFMINPRE:YUNIT? might return :WFMINPRE:YUNIT "V" indicating that the vertical units for the incoming waveform are volts.

WFMInpre:YZEro

Sets or returns the vertical offset of the incoming waveform in units specified by WFMInpre:YUNit. Variations in this number are analogous to changing the vertical offset of the waveform.

Group

Waveform

Syntax

`WFMInpre:YZEro <NR3>`

`WFMInpre:YZEro?`

Related commands

WFMInpre:YUNit on page 169, *WFMOutpre:YZEro?* on page 177

Arguments

`<NR3>` is offset, expressed in YUNits.

Examples

`WFMINPRE:YZERO 1.5E+0` specifies that the zero reference point for the incoming waveform is 1.5 V below the center of the data range (given that `WFMInpre:YUNit` is set to V).

`WFMINPRE:YZERO?` might return `:WFMINPRE:YZEro 7.5000E-6` indicating that the zero reference for the incoming waveform is 7.5 µV below the center of the data range (given that `WFMInpre:YUNit` is set to V).

WFMOutpre?

Returns waveform transmission and formatting settings for the waveform specified by the DATa:SOUrce command. Query only.

If the waveform specified by the DATa:SOUrce command is not displayed, the instrument returns only the waveform transmission parameters (BYT_Nr, BIT_Nr, ENCdg, BN_Fmt, BYT_Or).

Group

Waveform

Syntax

`WFMOutpre?`

Examples

`WFMOUTPRE?` might return the waveform formatting data as:

```
:WFMOUTPRE:BYT_NR 2;BIT_NR 16;ENCDG ASCII;BN_FMT RI;BYT_ORMSB;WFID "Ch1, DC
coupling, 100.0mV/div, 4.000us/div, 10000 points, Sample mode";NR_PT 2000;PT_FMT
Y;XUNIT "s";XINCR 4.0000E-9;XZERO - 20.0000E-6;PT_OFF 0;YUNIT "V";YMULT
15.6250E-6;YOFF :"6.4000E+3;YZERO 0.0000.
```

WFMOutpre:BIT_Nr

Sets and queries the number of bits per waveform point that outgoing waveforms contain, as specified by the DATa:SOUrce command. Changing the value of WFMOutpre:BIT_Nr also changes the values of WFMInpre:FILTERFreq and DATa:WIDth.

Group

Waveform

Syntax

WFMOutpre:BIT_Nr <NR1>

WFMOutpre:BIT_Nr?

Related commands

DATa:SOUrce on page 58, *DATa:WIDth* on page 60

Arguments

<NR1> is the number of bits per data point and can be 8 or 16.

Examples

WFMOUTPRE:BIT_NR 16 sets the number of bits per waveform point to 16 for outgoing waveforms.

WFMOUTPRE:BIT_NR? might return :WFMOUTPRE:BIT_NR 8 indicating that outgoing waveforms use 8 bits per waveform point.

WFMOutpre:BN_Fmt

Sets or returns the format of binary data for outgoing waveforms specified by the DATa:SOUrce command. Changing the value of WFMOutpre:BN_Fmt also changes the value of DATa:ENCdg.

Group

Waveform

Syntax

WFMOutpre:BN_Fmt {RI|RP}

WFMOutpre:BN_Fmt?

Arguments

RI specifies signed integer data point representation.

RP specifies positive integer data point representation.

Examples

WFMOUTPRE:BN_FMT RP specifies that outgoing waveform data will be in positive integer format.

WFMOUTPRE:BN_FMT? might return :WFMOUTPRE:BN_FMT RI indicating that the outgoing waveform data is currently in signed integer format.

WFMOutpre:BYT_Nr

Sets or queries the data width for the outgoing waveform specified by the DATa:SOUrce command. Changing WFMOutpre:BYT_Nr also changes WFMOutpre:BIT_Nr and DATa:WIDth.

Group

Waveform

Syntax

WFMOutpre:BYT_Nr <NR1>

WFMOutpre:BYT_Nr?

Related commands

DATa:SOUrce on page 58, *DATa:WIDth* on page 60, *WFMOutpre:BIT_Nr* on page 171

Arguments

<NR1> is the number of bytes per data point and can be 1 or 2.

Examples

WFMOUTPRE:BYT_NR 1 sets the number of bytes per outgoing waveform data point to 1, which is the default setting.

WFMOUTPRE:BYT_NR? might return :WFMOUTPRE:BYT_NR 2 indicating that there are 2 bytes per outgoing waveform data point.

WFMOutpre:ENCdg

Sets and queries the type of encoding for outgoing waveforms.

Group

Waveform

Syntax

WFMOutpre:ENCdg {ASCii|BINary}

WFMOutpre:ENCdg?

Related commands

WFMOutpre:BYT_Nr on page 172, *WFMOutpre:BIT_Nr* on page 171

Arguments

ASCii specifies that the outgoing data is to be in ASCII format. Waveforms will be sent as <NR1> numbers.

BINary specifies that outgoing data is to be in a binary format whose further specification is determined by WFMOutpre:BYT_Nr, WFMOutpre:BIT_Nr, WFMOutpre:BN_Fmt and WFMInpre:FILTERFreq.

Examples

WFMOutpre:ENCdg ASCii sets the encoding to ASCII.

WFMOutpre:ENCdg? might return WFMOutpre:ENCdg BINARY indicating the encoding is set to binary.

WFMOutpre:NR_Pt?

Returns the number of points for the DATa:SOUrce waveform that will be transmitted in response to a CURVe? query. The query command will timeout and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:NR_Pt?

Related commands

CURVe on page 55, *DATa* on page 57, *DATa:STARt* on page 58, *DATa:STOP* on page 59, *SAVe:WAVEform* on page 138, *SAVe:WAVEform:FILEFormat* on page 139, *WFMInpre:NR_Pt?* on page 166

Examples

WFMOUTPRE:NR_PT? might return :WFMOUTPRE:NR_PT 2000 indicating that there are 2000 data points to be sent.

WFMOutpre:RECOrdlength?

Returns the record length for the source waveform as specified by the DATa:SOUrce command. Query only.

Group

Waveform

Syntax

WFMOutpre:RECOrdlength?

Examples

WFMOUTPRE:RECORDLENGTH? might return :WFMOUTPRE:RECORDLENGTH 2000 indicating that 2000 is the source waveform record length.

WFMOutpre:WFId?

Returns a string describing several aspects of the acquisition parameters for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

`WFMOutpre:WFId?`

Related commands

DATa:SOUrce on page 58

Returns

<QString> comprises the following comma-separated fields:

`Source` The source identification string as it appears in the front-panel scale factor readouts.

`Coupling` A string describing the vertical coupling of the waveform.

`Vert Scale` A string containing the vertical scale factor of the unzoomed waveform. The numeric portion will always be four digits. The examples cover all known internal units.

`Horiz Scale` A string containing the horizontal scale factor of the unzoomed waveform. The numeric portion will always be four digits. The examples cover all known internal units.

`Record Length` A string containing the number of waveform points available in the entire record. The numeric portion is given as an integer.

`Acquisition Mode` A string describing the mode used to acquire the waveform.

Examples

`WFMOUTPRE:WFID?` might return `:WFMOUTPRE:WFID "Ch1, DC coupling,100.0mVolts/div,500.0µs/div, 1000 points, Sample mode"`.

WFMOutpre:XINcr?

Returns the horizontal point spacing in units of `WFMOutpre:XUNit` for the waveform specified by the `DATa:SOUrce` command. This value corresponds to the sampling interval. The query command will time out and an error will be generated if the waveform specified by `DATa:SOUrce` is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:XINcr?

Related commands

DATa:SOUrce on page 58, *WFMOutpre:XUNit?* on page 175

Examples

`WFMOUTPRE:XINCR?` might return `:WFMOUTPRE:XINCR 10.0000E-6` indicating that the horizontal sampling interval is 10 µs/point.

WFMOutpre:XUNit?

Returns the horizontal units for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:XUNit?

Related commands

DATa:SOUrce on page 58

Examples

`WFMOUTPRE:XUNIT?` might return `:WFMOUTPRE:XUNIT "HZ"` indicating that the horizontal units for the waveform are in Hertz.

WFMOutpre:XZEro?

Returns the time coordinate of the first point in the outgoing waveform. This value is in units of WFMOutpre:XUNit?. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

`WFMOutpre:XZEro?`

Related commands

DATa:SOUrce on page 58, *WFMOutpre:XUNit?* on page 175

Examples

`WFMOUTPRE:XZERO?` might return `:WFMOUTPRE:XZERO 5.6300E-9` indicating that the trigger occurred 5.63 ns before the first sample in the waveform record.

WFMOutpre:YMUlt?

Returns the vertical scale factor per digitizing level in units specified by WFMOutpre:YUNit for the waveform specified by the Returns the vertical scale factor per digitizing level in units specified by WFMOutpre:YUNit for the waveform specified by the DATa:SOUrce command. The query command will time out and an error is generated if the waveform specified by DATa:SOUrce is not turned on. command. The query command will time out and an error is generated if the waveform specified by DATa:SOUrce is not turned on. (Query Only)

Group

Waveform

Syntax

WFMOutpre:YMUlt?

Related commands

DATa:SOUrce on page 58, *WFMInpre:YMUlt* on page 168

Examples

WFMOUTPRE:YMULT? might return :WFMOUTPRE:YMULT 4.0000E-3 indicating that the vertical scale for the corresponding waveform is 100 mV/div (for 8-bit waveform data).

WFMOutpre:YOFf?

Returns the vertical position in digitizing levels for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:YOFf?

Related commands

DATa:SOUrce on page 58, *WFMOutpre:BYT_Nr* on page 172

Examples

WFMOUTPRE:YOFF? might return :WFMOUTPRE:YOFF -50.0000E+0 indicating that the position indicator for the waveform was 50 digitizing levels (2 divisions) below center screen (for 8-bit waveform data).

WFMOutpre:YUNit?

Returns the vertical units for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:YUNit?

Related commands

DATa:SOUrce on page 58

Examples

WFMOUTPRE:YUNIT? might return :WFMOUTPRE:YUNIT "dB" indicating that the vertical units for the waveform are measured in decibels.

WFMOutpre:YZEro?

Returns the vertical offset in units specified by WFMOutpre:YUNit? for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on. Query only.

Group

Waveform

Syntax

WFMOutpre:YZEro?

Related commands

DATa:SOUrce on page 58, *WFMOutpre:YUNit?* on page 177

Examples

WFMOUTPRE:YZERO? might return :WFMOUTPRE:YZERO -100.0000E-3 indicating that vertical offset is set to –100 mV.

Z commands

This section lists commands and queries that begin with the letter Z.

ZOOM?

Returns the current vertical and horizontal positioning and scaling of the display. Query only.

Group

Zoom

Syntax

`ZOOM?`

Examples

`ZOOM?` might return `:ZOOM:MODE 1; :ZOOM:ZOOM1:STATE 1;SCALE 20.0000E-9;POSITION 50.0000; FACTOR 10.0000; HORIZONTAL:POSITION 50.0000;SCALE 20.0000E-9.`

ZOOm{:MODe|:STATE}

Turns Zoom mode on or off. The Zoom mode query returns the current state of Zoom mode.

This command is equivalent to pressing the zoom button located on the front panel.

Group

Zoom

Syntax

`ZOOm{:MODe|:STATE} {ON|OFF|<NR1>}`

`ZOOm{:MODe|:STATE}?`

Arguments

`ON` turns on Zoom mode.

`OFF` turns off Zoom mode.

`<NR1>` = 0 turns off Zoom mode; any other value turns on Zoom mode.

Examples

`ZOOM:MODE OFF` turns off Zoom mode.

`ZOOM:MODE?` might return `:ZOOM:MODE 1` indicating that Zoom mode is currently turned on.

ZOOm:ZOOM1?

Returns the current horizontal positioning and scaling of the display. Query only.

Group

Zoom

Syntax

ZOOm:ZOOM1?

Examples

`ZOOM:ZOOM1?` might return `:ZOOM:ZOOM1:STATE 1;SCALE 20.0000E-9;POSITION 50.0000;FACTOR 10.0000;HORIZONTAL:POSITION 50.0000;SCALE 20.0000E-9`.

ZOOM:ZOOM1:FACtor

Queries the zoom factor of a particular zoom box. Query only.

Group

Zoom

Syntax

`ZOOM:ZOOM1:FACtor?`

Returns

`<NR1>` is the zoom factor of a zoom box.

Examples

`ZOOM:ZOOM1:FACtor?` might return `:ZOOM:ZOOM1:FACtor X5` indicating that the specified zoom factor is X5 of the acquired waveform.

ZOOM:ZOOM1:HORizontal:POSition

Sets or queries the horizontal position of a specified zoom box.

Group

Zoom

Syntax

`ZOOM:ZOOM1:HORizontal:POSition <NR1>`

`ZOOM:ZOOM1:HORizontal:POSition?`

Arguments

<NR1> is 1 to 100.00 and is the percent of the upper window that is to the left of the screen center, when the zoom factor is 1× or greater.

Examples

ZOOM:ZOOM1:HORizontal:POSition 50.00 sets the zoom reference pointer at 50% of the acquired waveform.

ZOOM:ZOOM1:HORIZONTAL:POSITION? might return :ZOOM1:HORIZONTAL:POSITION 50.00, indicating the reference pointer is at 50% of the acquired waveform.

ZOOM:ZOOM1:HORizontal:SCAle

Sets or queries the zoom horizontal scale for the specified zoom.

Group

Zoom

Syntax

ZOOm:ZOOM1:HORizontal:SCAle <NR3>

ZOOm:ZOOM1:HORizontal:SCAle?

Arguments

<NR3> is the amount of expansion in the horizontal direction and ranges from 1.0E-9 to 100.0.

Examples

ZOOM:ZOOM1:HORIZONTAL:SCALE 5 sets the horizontal scale to 5 seconds per division.

ZOOM:ZOOM2:HORIZONTAL:SCALE? might return :ZOOM2:HORIZONTAL:SCALE 1, indicating that the horizontal scale is 1 second per division.

ZOOm:ZOOM1:POSition

Sets or queries the horizontal position for the specified zoom.

Group

Zoom

Syntax

ZOOm:ZOOM1:POSition <NR3>

ZOOm:ZOOM1:POSition?

Arguments

<NR3> is a value from 0 to 100.00 and is the percent of the upper window that is to the left of screen center, when the zoom factor is 1× or greater

Examples

ZOOm:ZOOM1:POSition 20 sets the percent of the upper window that is to the left of screen center to 20%.

ZOOm:ZOOM1:POSition? might return :ZOOm:ZOOM1:POSition 25 indicating the percent of the upper window that is to the left of the screen center is 25%.

ZOOm:ZOOM1:SCAle

Sets or queries the zoom horizontal scale for the specified zoom.

Group

Zoom

Syntax

ZOOm:ZOOM1:SCAle <NR3>

ZOOm:ZOOM1:SCAle?

Arguments

<NR3> is the amount of expansion in the horizontal direction and ranges from 1.0E-9 to 100.0.

Examples

ZOOm:ZOOM1:SCAle 5.0 sets the horizontal expansion of the specified Zoom to 5.

ZOOm:ZOOM1:SCAle? might return ZOOm:ZOOM1:SCAle 5.0 indicating the zoom1 horizontal expansion is set to 5.

ZOOM:ZOOM1:STATE

Sets or queries the specified zoom on or off, where x is the integer representing the specified zoom window.

Group

Zoom

Syntax

ZOOm:ZOOM1:STATE {ON|OFF|<NR1>}

ZOOm:ZOOM1:STATE?

Arguments

ON turns the specified Zoom on.

OFF turns the specified Zoom off.

<NR1> = 0 disables the specified zoom; any other value enables the specified zoom

Examples

ZOOM:ZOOM1:STATE ON turns Zoom1 on.

`ZOOM:ZOOM1:STATE?` might return `:ZOOM:ZOOM1:STATE 1` indicating that Zoom1 is on.

Status and Events

The instrument provides a status and event reporting system for the GPIB, RS-232, and USB interfaces. This system informs you of certain significant events that occur within the instrument.

The instrument status reporting system consists of five 8-bit registers and two queues. This section describes these registers and components, and explains how the event handling system operates.

Registers

Overview

The registers in the event handling system fall into two functional groups:

- Status Registers contain information about the status of the instrument. They include the Standard Event Status Register (SESR).
- Enable Registers determine whether selected types of events are reported to the Status Registers and the Event Queue. They include the Device Event Status Enable Register (DESER), the Event Status Enable Register (ESER), and the Service Request Enable Register (SRER).

Status Registers

The Standard Event Status Register (SESR) and the Status Byte Register (SBR) record certain types of events that may occur while the instrument is in use. IEEE Std 488.2-1987 defines these registers.

Each bit in a Status Register records a particular type of event, such as an execution error or message available. When an event of a given type occurs, the instrument sets the bit that represents that type of event to a value of one. (You can disable bits so that they ignore events and remain at zero. See Enable Registers). Reading the status registers tells you what types of events have occurred.

The Standard Event Status Register (SESR)

The SESR records eight types of events that can occur within the instrument. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 3: The Standard Event Status Register (SESR)

Table 9: SESR bit functions

Bit	Function	
7 (MSB)	PON	`Power On.` Shows that the instrument was powered on. On completion, the diagnostic self tests also set this bit.
6	URQ	`User Request.` Indicates that an application event has occurred. *See note.
5	CME	`Command Error.` Shows that an error occurred while the instrument was parsing a command or query.
4	EXE	`Execution Error.` Shows that an error executing a command or query.

Table continued…

Bit	Function	
3	DDE	`Device Error.` Shows that a device error occurred.
2	QYE	`Query Error.` Either an attempt was made to read the Output Queue when no data was present or pending, or that data in the Output Queue was lost.
1	RQC	`Request Control.` This is not used.
0 (LSB)	OPC	`Operation Complete.` Shows that the operation is complete. This bit is set when all pending operations complete following an *OPC command.

The Status Byte Register (SBR)

Records whether output is available in the Output Queue, whether the instrument requests service, and whether the SESR has recorded any events.

Use a Serial Poll or the *STB? query to read the contents of the SBR. The bits in the SBR are set and cleared depending on the contents of the SESR, the Event Status Enable Register (ESER), and the Output Queue. When you use a Serial Poll to obtain the SBR, bit 6 is the RQS bit. When you use the *STB? query to obtain the SBR, bit 6 is the MSS bit. Reading the SBR does not clear the bits.

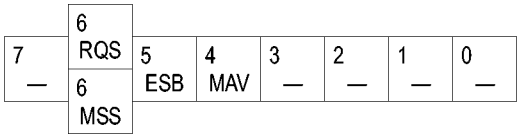


Figure 4: The Status Byte Register (SBR)

Table 10: SBR bit functions

Bit	Function	
7 (MSB)	----------	Not used.
6	RQS	`Request Service.` Obtained from a serial poll. Shows that the instrument requests service from the GPIB controller.
6	MSS	`Master Status Summary.` Obtained from *STB? query. Summarizes the ESB and MAV bits in the SBR.
5	ESB	`Event Status Bit.` Shows that status is enabled and present in the SESR.
4	MAV	`Message Available.` Shows that output is available in the Output Queue.
3	----------	Not used.
2	----------	Not used.
1–0	----------	Not used.

Enable Registers

DESER, ESER, and SRER allow you to select which events are reported to the Status Registers and the Event Queue. Each Enable Register acts as a filter to a Status Register (the DESER also acts as a filter to the Event Queue) and can prevent information from being recorded in the register or queue.

Each bit in an Enable Register corresponds to a bit in the Status Register it controls. In order for an event to be reported to a bit in the Status Register, the corresponding bit in the Enable Register must be set to one. If the bit in the Enable Register is set to zero, the event is not recorded.

Various commands set the bits in the Enable Registers. The Enable Registers and the commands used to set them are described below.

The Device Event Status Enable Register (DESER)

This register controls which types of events are reported to the SESR and the Event Queue. The bits in the DESER correspond to those in the SESR.

Use the DESE command to enable and disable the bits in the DESER. Use the DESE? query to read the DESER.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 5: The Device Event Status Enable Register (DESER)

The Event Status Enable Register (ESER)

This register controls which types of events are summarized by the Event Status Bit (ESB) in the SBR. Use the *ESE command to set the bits in the ESER. Use the *ESE? query to read it.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 6: The Event Status Enable Register (ESER)

The Service Request Enable Register (SRER)

This register controls which bits in the SBR generate a Service Request and are summarized by the Master Status Summary (MSS) bit.

Use the *SRE command to set the SRER. Use the *SRE? query to read the register. The RQS bit remains set to one until either the Status Byte Register is read with a Serial Poll or the MSS bit changes back to a zero.

7	6	5	4	3	2	1	0
—	—	ESB	MAV	—	—	—	—

Figure 7: The Service Request Enable Register (SRER)

*PSC Command

The *PSC command controls the Enable Registers contents at power-on. Sending *PSC 1 sets the Enable Registers at power on as follows:

- DESER 255 (equivalent to a DESe 255 command)
- ESER 0 (equivalent to an *ESE 0 command)
- SRER 0 (equivalent to an *SRE 0 command)

Sending *PSC 0 lets the Enable Registers maintain their values in nonvolatile memory through a power cycle.

Note: To enable the PON (Power On) event to generate a Service Request, send *PSC 0, use the DESe and *ESE commands to enable PON in the DESER and ESER, and use the *SRE command to enable bit 5 in the SRER. Subsequent power-on cycles will generate a Service Request.

Queues

The *PSC command controls the Enable Registers contents at power-on. Sending *PSC 1 sets the Enable Registers at power on as follows:

Output Queue

The instrument stores query responses in the Output Queue and empties this queue each time it receives a new command or query message after an <EOM>. The controller must read a query response before it sends the next command (or query) or it will lose responses to earlier queries.

CAUTION: When a controller sends a query, an <EOM>, and a second query, the instrument normally clears the first response and outputs the second while reporting a Query Error (QYE bit in the ESER) to indicate the lost response. A fast controller, however, may receive a part or all of the first response as well. To avoid this situation, the controller should always read the response immediately after sending any terminated query message or send a DCL (Device Clear) before sending the second query.

Event Queue

The Event Queue stores detailed information on up to 20 events. If than 20 events stack up in the Event Queue, the 20th event is replaced by event code 350, "Queue Overflow."

Read the Event Queue with the EVENT? query (which returns only the event number), with the EVMSG? query (which returns the event number and a text description of the event), or with the ALLEV? query (which returns all the event numbers with a description of the event). Reading an event removes it from the queue.

Before reading an event from the Event Queue, you must use the *ESR? query to read the summary of the event from the SESR. This makes the events summarized by the *ESR? read available to the EVENT? and EVMSG? queries, and empties the SESR.

Reading the SESR erases any events that were summarized by previous *ESR? reads but not read from the Event Queue. Events that follow an *ESR? read are put in the Event Queue but are not available until *ESR? is used again.

Event Handling Sequence

The following figure shows how to use the status and event handling system. In the explanation that follows, numbers in parentheses refer to numbers in the figure.

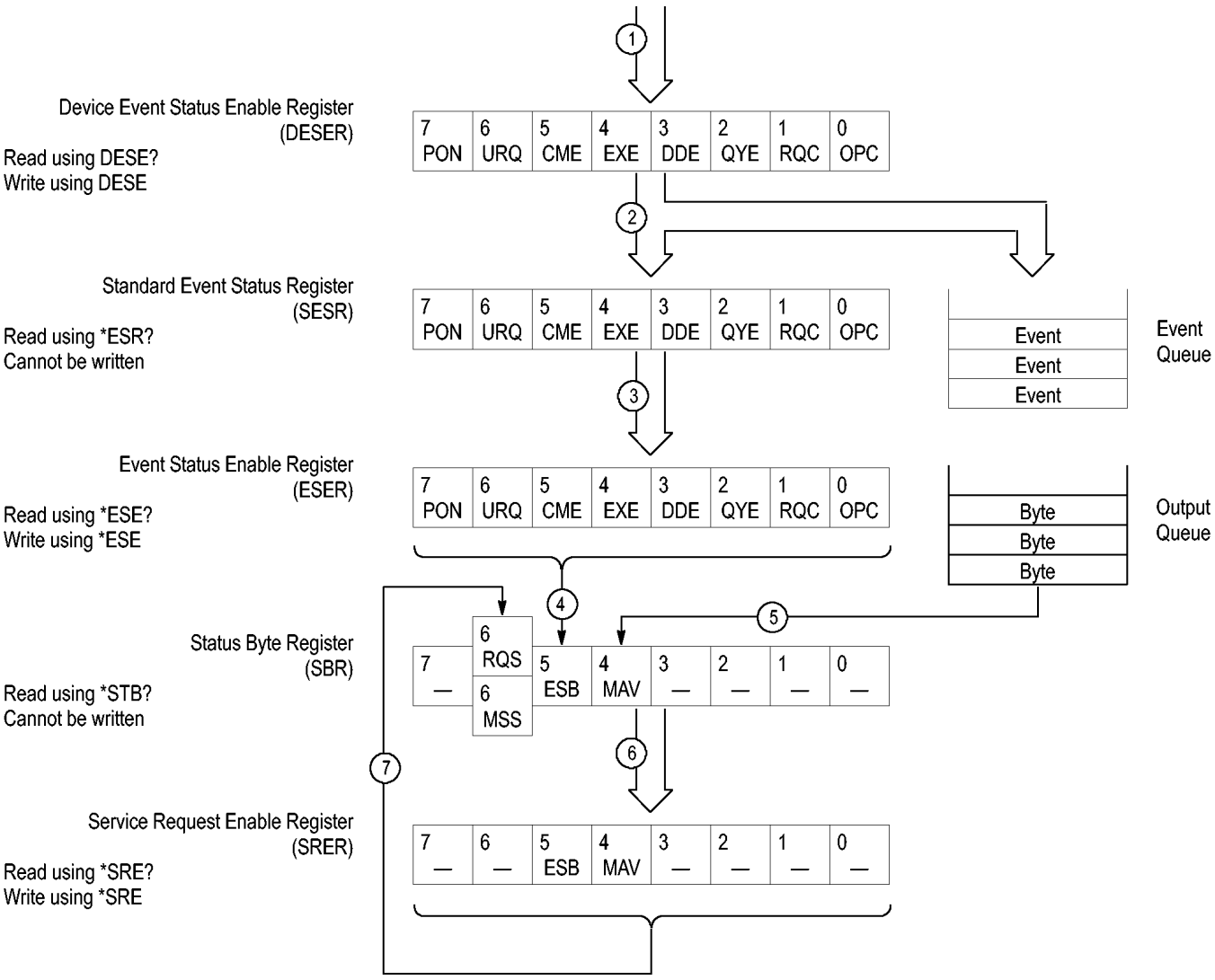


Figure 8: Status and Event Handling Process

When an event occurs, a signal is sent to the DESER (1). If that type of event is enabled in the DESER (that is, if the bit for that event type is set to 1), the appropriate bit in the SESR is set to one, and the event is recorded in the Event Queue (2). If the corresponding bit in the ESER is also enabled (3), then the ESB bit in the SBR is set to one (4).

When output is sent to the Output Queue, the MAV bit in the SBR is set to one (5).

When a bit in the SBR is set to one and the corresponding bit in the SRER is enabled (6), the MSS bit in the SBR is set to one and a service request is generated (7).

Synchronization Methods

Overview

Although most commands are completed almost immediately after being received by the instrument, some commands start a process that requires time. For example, once a single sequence acquisition command is executed, depending upon the applied signals and trigger settings, it may take an extended period of time before the acquisition is complete. Rather than remain idle while the operation is in

process, the instrument will continue processing other commands. This means that some operations will not be completed in the order that they were sent.

Sometimes the result of an operation depends on the result of an earlier operation. A first operation must complete before the next one is processed. The instrument status and event reporting system is designed to accommodate this process.

The Operation Complete (OPC) bit of the Standard Event Status Register (SESR) can be programmed to indicate when certain instrument operations have completed and, by setting the Event Status Enable Register (ESER) to report OPC in the Event Status Bit (ESB) of the Status Byte Register (SBR) and setting the Service Request Enable Register (SRER) to generate service request upon a positive transition of the ESB, a service request (SRQ) interrupt can be generated when certain operations complete as described in this section.

The following instrument operations can generate an OPC:

Table 11: Instrument operations that can generate OPC

Command	Conditions
ACQuire:STATE ON or ACQuire:STATE RUN	Only when ACQuire:STOPAfter is set to SEQuence
*CAL?	
CALibrate:CONTINUE	
CALibrate:FACtory	
CALIbrate:INTERNAL	
FACtory	
HARDCopy STARt	
RECAll:SETUp <file as quoted string>	
RECAll:WAVEform <file as quoted string>	
*RST	
SAVe:IMAGe <file as quoted string>	
SAVe:SETUp <file as quoted string>	
SAVe:WAVEform <file as quoted string>	

For example, a typical application might involve acquiring a single-sequence waveform and then taking a measurement on the acquired waveform. You could use the following command sequence to do this:

```
/** Set up conditional acquisition **/
ACQUIRE:STATE OFF
SELECT:CH1 ON
HORIZONTAL:RECORDLENGTH 1000
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

```
/** Acquire waveform data **/
ACQUIRE:STATE ON
/** Set up the measurement parameters **/
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT: IMMED:SOURCE CH 1
/** Take amplitude measurement **/
MEASUREMENT: MEAS1:VALUE?
```

The acquisition of the waveform requires extended processing time. It may not finish before the instrument takes an amplitude measurement (see the following figure). This can result in an incorrect amplitude value.

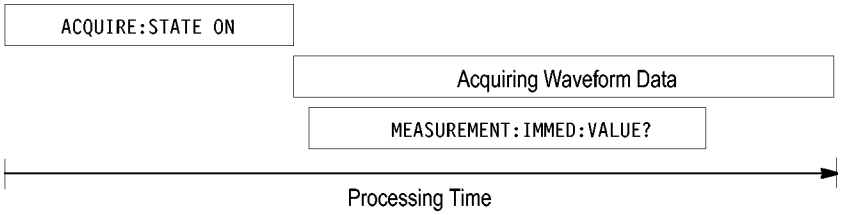


Figure 9: Command processing without using synchronization

To be sure the instrument completes waveform acquisition before taking the measurement on the acquired data, you can synchronize the program.

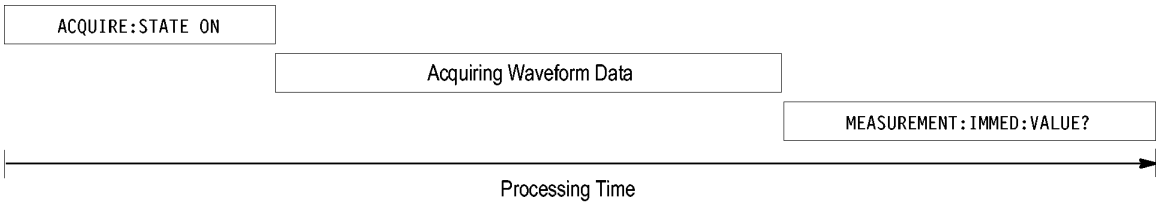


Figure 10: Processing sequence with synchronization

You can use four commands to synchronize the operation of the instrument with your application program: *WAI, BUSY, *OPC, and *OPC?

Using the *WAI Command

The *WAI command forces completion of previous commands that generate an OPC message. No commands after the *WAI are processed before the OPC message(s) are generated

The same command sequence using the *WAI command for synchronization looks like this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
```

```
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the measurement*/
*/
*WAI
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

The controller can continue to write commands to the input buffer of the instrument, but the commands will not be processed by the instrument until all in-process OPC operations are complete. If the input buffer becomes full, the controller will be unable to write commands to the buffer. This can cause a time-out.

Using the BUSY Query

The BUSY? query allows you to find out whether the instrument is busy processing a command that has an extended processing time such as single-sequence acquisition.

The same command sequence, using the BUSY? query for synchronization, looks like this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the measurement */
While BUSY? keep looping
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

This sequence lets you create your own wait loop rather than using the *WAI command. The BUSY? query helps you avoid time-outs caused by writing too many commands to the input buffer. The controller is still tied up though, and the repeated BUSY? query will result in bus traffic.

Using the *OPC Command

If the corresponding status registers are enabled, the *OPC command sets the OPC bit in the Standard Event Status Register (SESR) when an operation is complete. You achieve synchronization by using this command with either a serial poll or service request handler.

Serial Poll Method: Enable the OPC bit in the Device Event Status Enable Register (DESER) and the Event Status Enable Register (ESER) using the DESE and *ESE commands.

When the operation is complete, the OPC bit in the Standard Event Status Register (SESR) will be enabled and the Event Status Bit (ESB) in the Status Byte Register will be enabled.

The same command sequence using the *OPC command for synchronization with serial polling looks like this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Enable the status registers */
DESE 1
*ESE 1
*SRE 0
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the measurement.*/
*OPC
While serial poll = 0, keep looping
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

This technique requires less bus traffic than did looping on BUSY.

Service Request Method: Enable the OPC bit in the Device Event Status Enable Register (DESER) and the Event Status Enable Register (ESER) using the DESE and *ESE commands.

You can also enable service requests by setting the ESB bit in the Service Request Enable Register (SRER) using the *SRE command. When the operation is complete, the instrument will generate a Service Request.

The same command sequence using the *OPC command for synchronization looks like this

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Enable the status registers */
DESE 1
*ESE 1
*SRE 32
```

```
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the measurement*/
*OPC
```

The program can now do different tasks such as talk to other devices. The SRQ, when it comes, interrupts those tasks and returns control to this task.

```
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

Using the *OPC? Query

The *OPC? query places a 1 in the Output Queue once an operation that generates an OPC message is complete. The *OPC? query does not return until all pending OPC operations have completed. Therefore, your time-out must be set to a time at least if the longest expected time for the operations to complete.

The same command sequence using the *OPC? query for synchronization looks like this:

```
/* Set up single sequence acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the measurement*/
*OPC?
```

Wait for read from Output Queue.

```
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

This is the simplest approach. It requires no status handling or loops. However, you must set the controller time-out for longer than the acquisition operation.

Messages

The information contained in the topic tabs above covers all the programming interface messages the instrument generates in response to commands and queries.

For most messages, a secondary message from the instrument gives detail about the cause of the error or the meaning of the message. This message is part of the message string and is separated from the main message by a semicolon.

Each message is the result of an event. Each type of event sets a specific bit in the SESR and is controlled by the equivalent bit in the DESER. Thus, each message is associated with a specific SESR bit. In the message tables, the associated SESR bit is specified in the table title, with exceptions noted with the error message text.

No Event

The following table shows the messages when the system has no events or status to report. These have no associated SESR bit.

Table 12: No Event messages

Code	Message
0	No events to report; queue empty
1	No events to report; new events pending *ESR?

Command Error

The following table shows the command error messages generated by improper syntax. Check that the command is properly formed and that it follows the rules in the section on command Syntax.

Table 13: Command error messages (CME bit 5)

Code	Message
100	Command error
101	Invalid character
102	Syntax error
103	Invalid separator
104	Data type error
105	GET not allowed
108	Parameter not allowed
109	Missing parameter
110	Command header error

Table continued…

Code	Message
112	Program mnemonic too long
113	Undefined header
120	Numeric data error
121	Invalid character in numeric
123	Exponent too large
124	Too many digits
130	Suffix error
131	Invalid suffix
134	Suffix too long
140	Character data error
141	Invalid character data
144	Character data too long
150	String data error
151	Invalid string data
152	String data too long
160	Block data error
161	Invalid block data
170	Command expression error
171	Invalid expression

Execution Error

The following table lists the execution errors that are detected during execution of a command.

Table 14: Execution error messages (EXE bit 4)

Code	Message
200	Execution error

Table continued…

Code	Message
221	Settings conflict
222	Data out of range
224	Illegal parameter value
241	Hardware missing
250	Mass storage error
251	Missing mass storage
252	Missing media
253	Corrupt media
254	Media full
255	Directory full
256	File name not found
257	File name error
258	Media protected
259	File name too long
270	Hardcopy error
271	Hardcopy device not responding
272	Hardcopy is busy
273	Hardcopy aborted
274	Hardcopy configuration error
280	Program error
282	Insufficient network printer information
283	Network printer not responding
284	Network printer server not responding
286	Program run time error

Table continued…

Code	Message
287	Print server not found
2200	Measurement error, Measurement system error
2201	Measurement error, Zero period
2202	Measurement error, No period, second waveform
2203	Measurement error, No period, second waveform
2204	Measurement error, Low amplitude, second waveform
2205	Measurement error, Low amplitude, second waveform
2206	Measurement error, Invalid gate
2207	Measurement error, Measurement overflow
2208	Measurement error, No backward Mid Ref crossing
2209	Measurement error, No second Mid Ref crossing
2210	Measurement error, No Mid Ref crossing, second waveform
2211	Measurement error, No backward Mid Ref crossing
2212	Measurement error, No negative crossing
2213	Measurement error, No positive crossing
2214	Measurement error, No crossing, target waveform
2215	Measurement error, No crossing, second waveform
2216	Measurement error, No crossing, target waveform
2217	Measurement error, Constant waveform
2219	Measurement error, No valid edge - No arm sample
2220	Measurement error, No valid edge - No arm cross
2221	Measurement error, No valid edge - No trigger cross
2222	Measurement error, No valid edge - No second cross
2223	Measurement error, Waveform mismatch

Table continued…

Code	Message
2224	Measurement error, WAIT calculating
2225	Measurement error, No waveform to measure
2226	Measurement error, Null Waveform
2227	Measurement error, Positive and Negative Clipping
2228	Measurement error, Positive Clipping
2229	Measurement error, Negative Clipping
2230	Measurement error, High Ref < Low Ref
2231	Measurement error, No statistics available
2233	Requested waveform is temporarily unavailable
2235	Math error, invalid math description
2240	Invalid password
2241	Waveform requested is invalid
2244	Source waveform is not active
2245	Saveref error, selected channel is turned off
2250	Reference error, the reference waveform file is invalid
2253	Reference error, too many points received
2254	Reference error, too few points received
2259	File too big
2260	Calibration error
2270	Alias error
2271	Alias syntax error
2273	Illegal alias label
2276	Alias expansion error
2277	Alias redefinition not allowed

Table continued…

Code	Message
2278	Alias header not found
2285	TekSecure(R) Pass
2286	TekSecure(R) Fail
2301	Cursor error, Off screen
2302	Cursor error, Cursors are off
2303	Cursor error, Cursor source waveform is off
2500	Setup error, file does not look like a setup file
2501	Setup warning, could not recall all values from external setup
2620	Mask error, too few points received
2760	Mark limit reached
2761	No mark present
2762	Search copy failed

Device Error

The following table lists the device errors that can occur during instrument operation. These errors may indicate that the instrument needs repair.

Table 15: Device error messages (DDE bit 3)

Code	Message
310	System error
311	Memory error
312	PUD memory lost
314	Save/recall memory lost

System Event

The following table lists the system event messages. These messages are generated whenever certain system conditions occur.

Table 16: System event messages

Code	Message
400	Query event
401	Power on (PON bit 7 set)
402	Operation complete (OPC bit 0 set)
403	User request (URQ bit 6 set)
404	Power fail (DDE bit 3 set)
405	Request control
410	Query INTERRUPTED (QYE bit 2 set)
420	Query UNTERMINATED (QYE bit 2 set)
430	Query DEADLOCKED (QYE bit 2 set)
440	Query UNTERMINATED after indefinite response (QYE bit 2 set)
468	Knob/Keypad value changed
472	Application variable changed

Execution Warning

The following table lists warning messages that do not interrupt the flow of command execution. They also notify you of a possible unexpected results.

Table 17: Execution warning messages (EXE bit 4)

Code	Message
528	Parameter out of range
532	Curve data too long, Curve truncated
533	Curve error, Preamble values are inconsistent
540	Measurement warning, Uncertain edge
541	Measurement warning, Low signal amplitude
542	Measurement warning, Unstable histogram
543	Measurement warning, Low resolution

Table continued…

Code	Message
544	Measurement warning, Uncertain edge
545	Measurement warning, Invalid in minmax
546	Measurement warning, Need 3 edges
547	Measurement warning, Clipping positive/negative
548	Measurement warning, Clipping positive
549	Measurement warning, Clipping negative

Table 18: Execution warning messages (EXE bit 4)

Code	Message
540	Measurement warning
541	Measurement warning, Low signal amplitude
542	Measurement warning, Unstable histogram
543	Measurement warning, Low resolution
544	Measurement warning, Uncertain edge
545	Measurement warning, Invalid min max
546	Measurement warning, Need 3 edges
547	Measurement warning, Clipping positive/negative
548	Measurement warning, Clipping positive
549	Measurement warning, Clipping negative

Internal Warning

The following table shows internal errors that indicate an internal fault in the instrument.

Table 19: Internal warning messages

Code	Message
600	Internal warning

Programming Examples

The following series of commands and queries illustrate many of the most common commands and techniques.

To use these commands and queries over USB, you must use a program or routines that interface to the USBTMC driver on your PC. You can also use the PC Communications software that came on the CD with your instrument to get the same data without having to write programs. For operating information, you can launch the PC Communications software and refer to the online help.

To use these commands and queries over GPIB, you must use a program or routines that interface to the GPIB hardware in your computer. The software is usually supplied by the GPIB hardware manufacturer.

In these examples, data sent from the controller computer to the instrument is prefaced with the > symbol. Replies from the instrument have no preface.

> REM "Check for any messages, and clear them from the queue."

> *ESR?

128

> ALLEV ?

:ALLEV 401,"Power on; "

> REM "Set the instrument to the default state."

> FACTORY

> REM "Set the instrument parameters that differ from the defaults."

> CH1:VOLTS 2.0

> HOR:MAIN:SCALE 100e-6

> TRIG:MAIN:LEVEL 2.4

> REM "Start a single sequence acquisition."

> ACQUIRE:STOPAFTER SEQUENCE

> ACQUIRE:STATE ON

> REM "Wait for the acquisition to complete."

> REM "Note: your controller program time-out must be set long enough to handle the wait."

> *OPC?

1

> REM "Use the instrument built-in measurements to measure the waveform you acquired."

> MEASU:IMMED:TYPE MEAN

> MEASU:IMMED:VALUE?

:MEASUREMENT:IMMED:VALUE 2.4631931782E0

> REM "Be sure to use the *esr? query to check for measurement errors."

> MEASU:IMMED:TYPE FREQ

> MEASU:IMMED:VALUE?

:MEASUREMENT:IMMED:VALUE 9.9E37

> *ESR?

16

> ALLEV?

:ALLEV 2202,"Measurement error, No period found; "

> REM "Query out the waveform points, for later analysis on your controller computer." > data:encdg ascii

> CURVE?

:CURVE 7,6,5,5,5,6,6,6,8 [...]

> REM "Query out the parameters used for calculating the times and voltages of the waveform points."

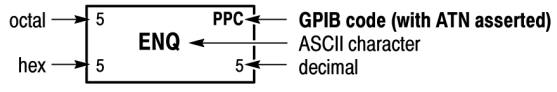
> WFMPRE?

:WFMPRE:BYT_NR 1;BIT_NR 8;ENCDG ASC;BN_FMT RP;BYT_OR MSB;NR_PT 2500; [...]

ASCII Code Chart

B7 B6 B5 BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 LA0 SP 20 32	60 LA16 0 30 48	100 TA0 @ 40 64	120 TA16 P 50 80	140 SA0 ` 60 96	160 SA16 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LL0 DC1 11 17	41 LA1 ! 21 33	61 LA17 1 31 49	101 TA1 A 41 65	121 TA17 Q 51 81	141 SA1 a 61 97	161 SA17 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 LA2 " 22 34	62 LA18 2 32 50	102 TA2 B 42 66	122 TA18 R 52 82	142 SA2 b 62 98	162 SA18 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 LA3 # 23 35	63 LA19 3 33 51	103 TA3 C 43 67	123 TA19 S 53 83	143 SA3 c 63 99	163 SA19 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 LA4 $ 24 36	64 LA20 4 34 52	104 TA4 D 44 68	124 TA20 T 54 84	144 SA4 d 64 100	164 SA20 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 LA5 % 25 37	65 LA21 5 35 53	105 TA5 E 45 69	125 TA21 U 55 85	145 SA5 e 65 101	165 SA21 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 LA6 & 26 38	66 LA22 6 36 54	106 TA6 F 46 70	126 TA22 V 56 86	146 SA6 f 66 102	166 SA22 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 LA7 ' 27 39	67 LA23 7 37 55	107 TA7 G 47 71	127 TA23 W 57 87	147 SA7 g 67 103	167 SA23 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 LA8 (28 40	70 LA24 8 38 56	110 TA8 H 48 72	130 TA24 X 58 88	150 SA8 h 68 104	170 SA24 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51 LA9) 29 41	71 LA25 9 39 57	111 TA9 I 49 73	131 TA25 Y 59 89	151 SA9 i 69 105	171 SA25 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 LA10 * 2A 42	72 LA26 : 3A 58	112 TA10 J 4A 74	132 TA26 Z 5A 90	152 SA10 j 6A 106	172 SA26 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 LA11 + 2B 43	73 LA27 ; 3B 59	113 TA11 K 4B 75	133 TA27 [5B 91	153 SA11 k 6B 107	173 SA27 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 LA12 , 2C 44	74 LA28 < 3C 60	114 TA12 L 4C 76	134 TA28 \ 5C 92	154 SA12 l 6C 108	174 SA28 ¦ 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 LA13 – 2D 45	75 LA29 = 3D 61	115 TA13 M 4D 77	135 TA29] 5D 93	155 SA13 m 6D 109	175 SA29 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 LA14 . 2E 46	76 LA30 > 3E 62	116 TA14 N 4E 78	136 TA30 ^ 5E 94	156 SA14 n 6E 110	176 SA30 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 LA15 / 2F 47	77 UNL ? 3F 63	117 TA15 O 4F 79	137 UNT _ 5F 95	157 SA15 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY



Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Factory setup

The following listing is the instrument response to the concatenated command FACtory;SET. This response describes the factory default setup in detail. (Carriage returns have been inserted for clarity.)

Items enclosed in ( ) parentheses are returned by the SET? query response, but are not changed by the FACtory command.

TBS1000C Series Oscilloscopes

:HEADER 1;(:VERBOSE 1;)

:DATA:ENCDG RIBINARY;DESTINATION REFA;SOURCE CH1;START 1;STOP 2500;WIDTH 1;

(:LOCK NONE;)

:DISPLAY:FORMAT YT;STYLE VECTORS;PERSISTENCE 0;(CONTRAST 50);(INVERT OFF);

:ACQUIRE:MODE SAMPLE;NUMAVG 16;STATE 1;STOPAFTER RUNSTOP;

:CH1:PROBE 10;SCALE 1.0E0;POSITION 0.0E0;COUPLING DC;BANDWIDTH OFF;INVERT OFF;

:CH2:PROBE 10;SCALE 1.0E0;POSITION 0.0E0;COUPLING DC;BANDWIDTH OFF;INVERT OFF;

:HORIZONTAL:VIEW MAIN;MAIN:SCALE 5.0E-4;POSITION 0.0E0;

:HORIZONTAL:DELAY:SCALE 5.0E-5;POSITION 0.0E0;

:TRIGGER:MAIN:MODE AUTO;TYPE EDGE;HOLDOFF:VALUE 5.0E-7;

:TRIGGER:MAIN:EDGE:SOURCE CH1;COUPLING DC;SLOPE RISE;

:TRIGGER:MAIN:VIDEO:SOURCE CH1;SYNC LINE;POLARITY NORMAL;LINE 1;STANDARD NTSC;

:TRIGGER:MAIN:PULSE:SOURCE CH1;WIDTH:POLARITY POSITIVE;WHEN EQUAL;WIDTH 1.0E-3;

:TRIGGER:MAIN:LEVEL 0.0E0;

:SELECT:CH1 1;CH2 0;MATH 0;REFA 0;REFB 0;REFC 0;REFD 0;

:CURSOR:FUNCTION OFF;SELECT:SOURCE CH1;

:CURSOR:VBARS:UNITS SECONDS;POSITION1 -2.0E-3;POSITION2 2.0E-3;

:CURSOR:HBARS:POSITION1 -3.2E0;POSITION2 3.2E0;

:MEASUREMENT:MEAS1:TYPE NONE;SOURCE CH1;

:MEASUREMENT:MEAS2:TYPE NONE;SOURCE CH1;

:MEASUREMENT:MEAS3:TYPE NONE;SOURCE CH1;

:MEASUREMENT:MEAS4:TYPE NONE;SOURCE CH1;

:MEASUREMENT:MEAS5:TYPE NONE;SOURCE CH1;

:MEASUREMENT:IMMED:TYPE PERIOD;SOURCE CH1;

:MATH:DEFINE "CH1 - CH2";FFT:HORIZONTAL:POSITION 5.0E1;SCALE 1.0E0;

:MATH:FFT:VERTICAL:POSITION 0.0E0;SCALE 1.0E0;

(<SAVE:IMAGE:FILEFORMAT BMP;>)

(:LANGUAGE ENGLISH)

Reserved words

The following words are reserved for the instrument.

*CAL, *CLS, *DDT, *ESE, *ESR, *IDN, *LRN, *OPC, *PSC, *RCL, *RST, *SAV, *SRE, *STB, *TRG, *TST, *WAI, A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, ABOrt, AC, ACLINE, ACQuire, ALL, ALLEv, ASC, ASCIi, AUTO, AUTOMATIC, AUTORange, AUTOSet, AVErage, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9, BANdwidth, BATTERIES, BAUd, BIN, BIT_Nr, BMP, BN_Fmt, BOTH, BRIGHTness, BUBBLEJet, BUSY, BUTTON, BUTTONLIGHT, BYT_Nr, BYT_Or, Block, CALibrate, CARD, CENtronics, CH1, CH1CH2, CH2, CM10BY15, CM13BY18, CM15BY21, CM18BY24, CM6BY8, CM7BY10, CM9BY13, COMpare, CONDUCTION, CONTINUE, CONTINUOUS, CONTRast, COUPling, CR, CRLf, CRMs, CURRENTPRObe, CURSor, CURSORRms, CURVe, CWD, DATALOGging, DATE, DATEPRINT, DATa, DC, DCLIne, DEF, DEFINE, DEFLT, DEFault, DELay, DELEte, DELTa, DELay, DESE, DESKJet, DESTination, DIAg, DIR, DISplay, DOTs, DPU3445, DPU411, DPU412, DRAFT, DURAtion, E, EDGE, ENAble, ENCdg, ENGLish, ENV, EPSC60, EPSC80, EPSIMAGE, EPSOn, EQual, ERRLOG, EVEN, EVENT, EVMsg, EVQty, EXECute, EXT, EXT10, EXT5, FACtory, FALL, FALLINGedge, FASTPHOTO, FFT, FIELD, FILEFormat, FILESystem, FINE, FIRST, FLAg, FORCe, FORMat, FREESpace, FRENch, FREQuency, FREquency, FUNCtion, GASgauge, GERMan, GND, GPIb, HAGAKIPC, HAGAKIPCARD, HARDCopy, HARDFlagging, HARmonics, HBArs, HDELTa, HDR, HEADer, HERtz, HFRej, HOLDOff, HORizontal, HRMS, ID, IDPRINT, IMAGESIZE, IMAge, IMMed, IN11BY17, IN2P5BY3P25, IN4BY6, IN8BY10, INDEX, INF, INIT, INKSaver, INTERLEAF, INTERNAL, INVERT, INVert, INside, ITALian, JAPAnese, JOULES, JPEG, JPG, KOREan, L, L2, L4, LANGuage, LANdscape, LASERJet, LAYout, LETTER, LEVELS, LEVel, LF, LFCr, LFRej, LIMit, LINE, LINENum, LOCk, LOG, LSB, MAIn, MANUAL, MATH, MAXImum, MEAN, MEASUrement, MINImum, MKDir, MM100BY150, MM54BY86, MODe, MSB, MULTICYcle, N, NEGAtive, NEXT, NOISERej, NONE, NONe, NORMal, NOTEqual, NRMAL, NR_Pt, NTSc, NUMACq, NUMAVg, NWIdth, ODD, OFF, ON, OUTside, PAL, PAPERSIZE, PAPERTYPE, PARity, PCX, PEAKdetect, PERCent, PERIod, PERSistence, PFPHASE, PHAse, PHOTO, PICTBridge, PK2pk, PLAIN, POLarity, PORT, PORTRait, PORTUguese, POSITIVe, POSition, POWERFACTOR, POWer, POWerANALYSIS, PRESENt, PRINTQUAL, PRINTS, PRObe, PT_Fmt, PT_Off, PULse, PWIdth, RECAll, RECOrdlength, REFx, REM, REName, RESUlt, RI, RIBinary, RISINGedge, RISe, RLE, RMDir, RMS, ROLL100MM, ROLL127MM, ROLL210MM, ROLL89MM, RP, RPBinary, RS232, RUN, RUNSTop, SAMple, SAVESAll, SAVESImage, SAVE, SAVEIMAge, SAVEWFM, SCAle, SECOnds, SECdiv, SELect, SEQuence, SET, SETLevel, SETTings, SETUp, SHOW, SIGNAL, SIMPlifiedchinese, SINGLECYcle, SLOpe, SOFTFlagging, SOURCE, SOURCE1, SOURCE2, SOUrce, SOUrces, SPANish, SRIbinary, SRPbinary, STANDard, STARt, STATE, STATUS, STOP, STOPAfter, STYle, SWLoss, SYNC, TARget, TEMPLate, TERMinator, THDF, THDR, THINKjet, TIFF, TIME, TIMe, TOFFEND, TOFFSTART, TOLerance, TONEND, TONSTART, TOTAL, TRADitionalchinese, TRANsmit, TRIGger, TRUEPOWER, TURNOFF, TURNON, TYPe, UNIts, UNLock, USB, VALue, VAR, VBArs, VDELTa, VECtors, VERBose, VERtical, VIDeo, VIEW, VIOLation, VOLts, VSAT, WATTS, WAVEform, WAVEFORMANALYSIS, WAVEform, WAVFrm, WFCREST, WFCYCRMS, WFFREQ, WFId, WFMPre, WHEN, WIDth, WINDOW, XINcr, XUNit, XY, XZEro, Y, YMULt, YOFf, YT, YUNit, YZEro, ZONE,

Glossary terms

ASCII	Acronym for the American Standard Code for Information Interchange. Controllers transmit commands to the digitizing instrument using ASCII character encoding.
Address	A 7-bit code that identifies an instrument on the communication bus. The digitizing instrument must have a unique address for the controller to recognize and transmit commands to it.
Backus-Naur Form (BNF)	A standard notation system for command syntax. The syntax in this manual use BNF notation.
Controller	A computer or other device that sends commands to and accepts responses from the digitizing instrument.
EOI	A mnemonic referring to the control line "End or Identify" on the GPIB interface bus. One of the two possible end-of-message terminators.
EOM	A generic acronym referring to the end-of-message terminator. For GPIB, the end-of-message terminator is either an EOI or the ASCII code for line feed (LF). For USB, the end-of-message terminator is the EOM bit in a USBTMC message.
GPIB Address	When communicating with an instrument using a TEK-USB-488 adapter, you can set a unique GPIB address for the instrument in the UTILITY" Options " GPIB Setup option.
IEEE	Acronym for the Institute of Electrical and Electronics Engineers.
RS-232	A serial, full-duplex, asynchronous communication port that follows ANSI/EIA/TIA-562-1989[1], ANSI/EIA/TIA-574-1990[2], and CCITT V.24-1989[3] standards.
Serial Poll	A device (such as an instrument) on the GPIB bus can request service from the GPIB Controller by asserting the GPIB SRQ line (a Hardware line that is only present on the GPIB communications bus). A device on the USB bus can request service from the host by sending an SRQ packet on the Interrupt-IN endpoint. When a controller or a USB host acknowledges the SRQ, it "serial polls" each open device on the bus to determine which device on the bus requested service. Any device requesting service returns a status byte with bit 6 set and then unasserts the SRQ line (GPIB only). Devices not requiring service return a status byte with bit 6 cleared.
USB	An acronym for Universal Serial Bus.
USBTMC	An acronym for USB Test and Measurement Class.
USB488	The USBTMC subclass specification that implements an IEEE488-like interface over USB.

Index

Special Characters

*CAL? *33*
*CLS *46*
*ESE *69*
*ESR? *70*
*IDN? *100*
*LRN? *103*
*OPC *128*
*PSC *129*
*RCL *130*
*RST *134*
*SAV *135*
*SRE *143*
*STB? *144*
*WAI *163*

A

ACQuire:MAXSamplerate *22*
ACQuire:MODe *22*
ACQuire:NUMACq? *23*
ACQuire:NUMAVg *24*
ACQuire:STATE *24*
ACQuire:STOPAfter *25*
ACQuire? *22*
ALIas *26*
ALIas:CATalog? *26*
ALIas:DEFine *27*
ALIas:DELEte *27*
ALIas:DELEte:ALL *28*
ALIas:DELEte[:NAMe] *28*
ALIas[:STATE] *29*
ALLEv? *29*
Arguments *19*
ASCII *14*
AUTOSet *30*
AUTOSet:ENABLE *30*

B

BNF (Backus Naur form) *14*
BUSY? *32*

C

CALibrate:INTERNAL *33*
CALibrate:INTERNal:STARt *34*
CALibrate:INTERNal:STATus? *34*
CALibrate:RESults:SPC? *35*
CALibrate:RESults? *35*
CH<x>:AMPSVIAVOLTs :Factor *37*
CH<x>:AMPSVIAVOLTs:ENABLe *36*
CH<x>:BANdwidth *37*
CH<x>:COUPling *38*
CH<x>:DESKew *38*
CH<x>:INVert *39*
CH<x>:LABel *39*
CH<x>:OFFSet *40*
CH<x>:POSition *40*
CH<x>:PRObe *41*
CH<x>:PRObe:GAIN *41*
CH<x>:PRObe:ID:SERnumber? *42*
CH<x>:PRObe:ID:TYPE? *43*
CH<x>:PRObe:ID? *42*
CH<x>:PRObe:SIGnal *43*
CH<x>:PRObe:UNIts? *43*
CH<x>:SCAle *44*
CH<x>:VOLts *45*
CH<x>:YUNit *45*
CH<x>? *36*
CLEARMenu *46*
Command
  syntax:BNF (Backus Naur form) *14*
Command and Query Structure *14*
Command syntax
  BNF (Backus Naur form) *14*
CURSor:ENABLE *47*
CURSor:FUNCtion *48*
CURSor:HBArs:DELTa? *49*
CURSor:HBArs:POSITION<x> *49*
CURSor:HBArs:UNIts *50*
CURSor:HBArs? *48*
CURSor:VBArs:DELTa? *52*
CURSor:VBArs:HPOS<x>? *53*
CURSor:VBArs:POSITION<x> *53*
CURSor:VBArs:UNIts *54*
CURSor:VBArs:VDELTa? *54*
CURSor:VBArs? *51*
CURSor? *47*
CURVe *55*

D

DATa *57*
DATa:DESTination *57*
DATa:SOUrce *58*
DATa:STARt *58*
DATa:STOP *59*
DATa:WIDth *60*
DATE *60*
DESE *61*
DIAg:FAN *62*
DIAg:LOOP:OPTion *62*
DIAg:RESUlt:FLAg? *63*
DIAg:RESUlt:LOG? *64*
DIAg:SELect *64*
DIAg:SELect:<function> *65*
DIAg:STATE *65*
DIAg:TEMPVAL *66*

DISplay:GRAticule 66
DISplay:INTENSITy:BACKLight 66
DISplay:PERSistence:STATe 67
DISplay:PERSistence:VALUe 67

E

ERRLOG:FIRST? 69
ERRLOG:NEXT? 69
Event handling 183
EVENT? 70
EVMsg? 71
EVQty? 72
Example programming 202
Examples
    Programming 202

F

FACtory 73
Factory setup
    detailed description 204
FEAEN:PASSWORD 73
FFT:HORizontal:POSition 74
FFT:HORizontal:SCAle 75
FFT:SOURce 75
FFT:SRCWFM 76
FFT:VERTical:POSition 76
FFT:VERTical:SCAle 77
FFT:VERTical:UNIts 77
FFT:VType 77
FFT:WINdow 78
FFT? 74
FILESystem:CWD 79
FILESystem:DELEte 79
FILESystem:DIR? 80
FILESystem:FORMat 80
FILESystem:FREESpace? 81
FILESystem:MKDir 81
FILESystem:READFile 82
FILESystem:REName 82
FILESystem:RMDir 83
FILESystem:WRITEFile 84
FILESystem? 78
FPAnel:PRESS 84
FPAnel:TURN 85

H

HDR 87
HEADer 87
HELPevery:ACQuire 87
HELPevery:ALL 88
HELPevery:CURsor 88
HELPevery:FFT 89
HELPevery:MATH 89
HELPevery:MEASUrement 90
HELPevery:REFerence 90
HELPevery:TRIGger 90
HELPevery:UTIlity 91
HELPevery:VERtical 91
HORizontal:ACQLENGTH 92
HORizontal:DELay:POSition 93
HORizontal:DIVisions 93
HORizontal:MAIn:DELay:MODe 94
HORizontal:MAIn:DELay:STATe 94
HORizontal:MAIn:DELay:TIMe 95
HORizontal:MAIn:SAMPLERate 95
HORizontal:MAIn:SECdiv 96
HORizontal:MAIn:UNIts:STRing 97
HORizontal:PREViewstate 97
HORizontal:RECOrdlength 98
HORizontal:RECOrdlength:Auto 98
HORizontal:RESOlution 98
HORizontal:ROLL 99
HORizontal:SCALe 96
HORizontal:TRIGger:POSition 99
HORizontal? 92

I

ID? 100
IEEE Std. 488.2-1987 14

L

LANGuage 102
LOCk 102

M

MATH:DEFINE 104
MATH:HORizontal:POSition 105
MATH:HORizontal:SCALe 105
MATH:HORizontal:UNIts 106
MATH:LABel 106
MATH:VERtical:POSition 107
MATH:VERtical:SCAle 107
MATH:VERTical:UNIts 108
MATH? 104
MEASUrement:CLEARSNAPSHOT 109
MEASUrement:ENABLE 109
MEASUrement:GATing 110
MEASUrement:IMMed:DELay:EDGE<x> 111
MEASUrement:IMMed:DELay? 111
MEASUrement:IMMed:SOUrce1 111
MEASUrement:IMMed:SOUrce2 112
MEASUrement:IMMed:TYPe 113
MEASUrement:IMMed:UNIts? 114
MEASUrement:IMMed:VALue? 115
MEASUrement:IMMed? 110
MEASUrement:MEAS<x>:DELay:EDGE<x> 116
MEASUrement:MEAS<x>:DELay? 116
MEASUrement:MEAS<x>:SOUrce1 117
MEASUrement:MEAS<x>:SOUrce2 117
MEASUrement:MEAS<x>:STATE 118
MEASUrement:MEAS<x>:TYPe 118
MEASUrement:MEAS<x>:UNIts? 120

MEASUrement:MEAS<x>:VALue? *121*
MEASUrement:MEAS<x>? *115*
MEASUrement:REFLevel:ABSolute:LOW *122*
MEASUrement:REFLevel:ABSolute:MID1 *123*
MEASUrement:REFLevel:ABSolute:MID2 *123*
MEASUrement:REFLevel:METHod *124*
MEASUrement:REFLevel:PERCent:HIGH *124*
MEASUrement:REFLevel:PERCent:LOW *125*
MEASUrement:REFLevel:PERCent:MID1 *126*
MEASUrement:REFLevel:PERCent:MID2 *126*
MEASUrement:REFLevel? *122*
MEASUrement:SNAPSHOT *127*
MEASUrement:SOUrceSNAPShot *127*
MEASUrement? *108*
Message
handling *183*
Mnemonics *18*

P

Programming
examples *202*
Programming examples *202*

R

RECAll:SETUp *130*
RECAll:WAVEForm *131*
REF<x>:DATE *132*
REF<x>:HORizontal:DELay:TIMe? *132*
REF<x>:HORizontal:SCAle? *133*
REF<x>:POSition? *133*
REF<x>:TIMe? *132*
REF<x>:VERTical:POSition? *133*
REF<x>:VERTical:SCAle? *134*
REF<x>? *131*

S

SAVe:IMAge *136*
SAVe:IMAge:FILEFormat *136*
SAVe:IMAGe:LAYout *137*
SAVe:SETUp *137*
SAVe:WAVEform *138*
SAVe:WAVEform:FILEFormat *139*
SELect:CH<x> *139*
SELect:CONTROl *140*
SELect:FFT *140*
SELect:MATH *141*
SELect:REF<x> *141*
SET? *142*
SETUP<x>:DATE? *142*
SETUP<x>:TIME? *143*
Setups
factory:TBS2000 *204*
Status *183*
Syntax
BNF (Backus Naur form) *14*

T

TEKSecure *145*
TIMe *145*
TRIGger *146*
TRIGger:A *146*
TRIGger:A:EDGE:COUPling *147*
TRIGger:A:EDGE:SLOpe *148*
TRIGger:A:EDGE:SOUrce *148*
TRIGger:A:EDGE? *147*
TRIGger:A:HOLDOff:TIMe *149*
TRIGger:A:HOLDOff? *149*
TRIGger:A:LEVel *150*
TRIGger:A:LEVel:CH<x> *150*
TRIGger:A:LOWerthreshold:CH<x> *151*
TRIGger:A:MODe *152*
TRIGger:A:PULse:CLAss *153*
TRIGger:A:PULse:SOUrce *154*
TRIGger:A:PULse:WIDth:POLarity *154*
TRIGger:A:PULse:WIDth:WHEN *155*
TRIGger:A:PULse:WIDth:WIDth *155*
TRIGger:A:PULse:WIDth? *153*
TRIGger:A:PULse? *152*
TRIGger:A:RUNT *156*
TRIGger:A:RUNT:POLarity *156*
TRIGger:A:RUNT:SOUrce *157*
TRIGger:A:RUNT:WHEn *157*
TRIGger:A:RUNT:WIDth *158*
TRIGger:A:TYPe *158*
TRIGger:A:UPPerthreshold:CH<x> *159*
TRIGger:FREQuency? *160*
TRIGger:STATE? *160*

U

UNLock *161*

V

VERBose *162*

W

WAVFrm? *163*
WFMInpre:BIT_Nr *164*
WFMInpre:BYT_Nr *165*
WFMInpre:ENCdg *165*
WFMInpre:NR_Pt? *166*
WFMInpre:XINcr *166*
WFMInpre:XUNit *167*
WFMInpre:XZEro *167*
WFMInpre:YMULt *168*
WFMInpre:YOFf *168*
WFMInpre:YUNit *169*
WFMInpre:YZEro *170*
WFMInpre? *164*
WFMOutpre:BIT_Nr *171*
WFMOutpre:BN_Fmt *171*
WFMOutpre:BYT_Nr *172*

WFMOutpre:ENCdg *172*
WFMOutpre:NR_Pt? *173*
WFMOutpre:RECOrdlength? *173*
WFMOutpre:WFId? *174*
WFMOutpre:XINcr? *174*
WFMOutpre:XUNit? *175*
WFMOutpre:XZEro? *175*
WFMOutpre:YMUlt? *176*
WFMOutpre:YOFf? *176*
WFMOutpre:YUNit? *177*
WFMOutpre:YZEro? *177*
WFMOutpre? *170*

Z

ZOOm:ZOOM1 *179*
ZOOM:ZOOM1:FACtor *179*
ZOOM:ZOOM1:HORizontal:POSition *179*
ZOOM:ZOOM1:HORizontal:SCAle *180*
ZOOm:ZOOM1:POSition *180*
ZOOm:ZOOM1:SCAle *181*
ZOOM:ZOOM1:STATE *181*
ZOOM? *178*
ZOOm{:MODe|:STATE} *178*