

# ***SR1 Audio Analyzer***

---

## **GPIB Programming Manual**



---

Revision 1.0.0  
October, 2010

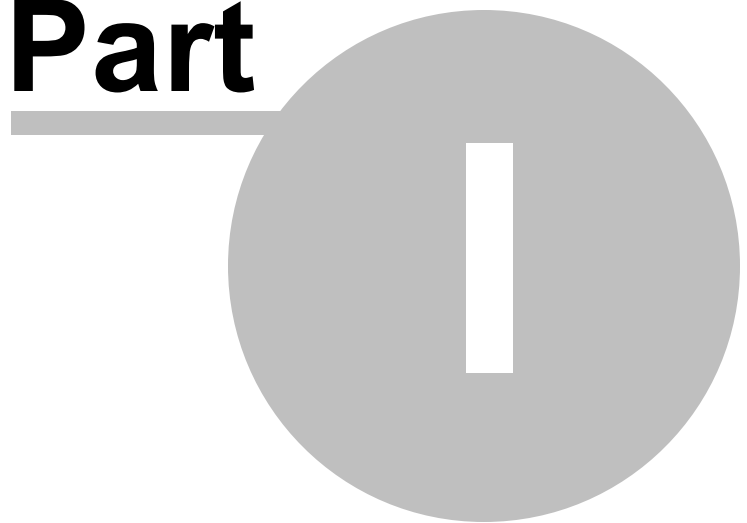
# Table of Contents

Foreword	0
<b>Part I Introduction</b>	<b>5</b>
1 Revision History.....	6
<b>Part II GPIB</b>	<b>8</b>
1 Using GPIB Commands.....	8
2 GPIB Status Model.....	14
3 GPIB Command Reference.....	17
<b>GPIB Common Commands</b> .....	<b>17</b>
<b>Form Commands</b> .....	<b>25</b>
<b>Analog Inputs</b> .....	<b>27</b>
<b>Analog Input Channel</b> .....	<b>28</b>
<b>Digital I/O</b> .....	<b>32</b>
<b>Sweep</b> .....	<b>93</b>
Sweep Source.....	109
Sweep Settling.....	126
<b>Analyzer</b> .....	<b>130</b>
Analyzer Trigger.....	141
FFT1 Analyzer.....	146
FFT2 Analyzer.....	157
Time Domain Detector.....	172
THD Analyzer.....	182
IMD Analyzer.....	189
Multitone Analyzer.....	193
Jitter Analyzer.....	206
Histogram Analyzer.....	223
<b>Multitone Configuration</b> .....	<b>231</b>
ToneA .....	242
ToneB .....	245
<b>Analyzer References</b> .....	<b>248</b>
<b>Digitizer</b> .....	<b>254</b>
<b>Clock References</b> .....	<b>269</b>
<b>Monitors</b> .....	<b>277</b>
<b>Instrument</b> .....	<b>282</b>
<b>Preferences</b> .....	<b>318</b>
<b>Scripting</b> .....	<b>347</b>
<b>Displays</b> .....	<b>354</b>
Graph .....	358
Graph Trace .....	372
Graph Cursor .....	388
Graph Limit .....	401
Digitizer Display.....	406
Digitizer Display Time Record.....	412
Digitizer Display Cursor.....	420
Digitizer Display Probability.....	425
Digitizer Display Spectrum.....	435
Digitizer Display Eye Diagram.....	446

Eye Diagram Limits .....	458
BarChart .....	464
<b>Event Manager .....</b>	<b>479</b>
<b>Switcher Configuration .....</b>	<b>490</b>
<b>Analog Generator .....</b>	<b>496</b>
Analog Generator Channel.....	505
Sine .....	509
Lo Distortion Sine.....	512
Phased Sine .....	515
Sync Burst Sine.....	518
Noise .....	523
USASI Noise .....	527
MLS Noise .....	529
Ramp .....	532
Square .....	535
IMD .....	537
Arbitrary .....	541
FFT Chirp .....	544
Log-Sine Chirp .....	548
Multitone .....	551
Polarity .....	554
Constant (DC) .....	556
<b>Digital Generator .....</b>	<b>558</b>
Digital Generator Channel.....	566
Sine .....	570
Phased Sine .....	573
Noise .....	576
USASI Noise .....	580
MLS Noise .....	582
Ramp .....	585
Chirp .....	588
Log-Sine Chirp .....	592
Arbitrary .....	595
Square .....	598
IMD .....	601
MultiTone .....	605
Polarity .....	608
Count .....	610
Digital Constant.....	614
Rotate .....	616
JTest .....	618
Stairstep .....	619
<b>Quick Measurements .....</b>	<b>621</b>
Level .....	634
Reference.....	645
SNR .....	649
THD+N .....	659
Distortion.....	670
IMD .....	681
Crosstalk.....	689
Frequency Response.....	700
Input/Output Phase .....	708
InterChannel Phase.....	716

# ***Introduction***

**Part**



# 1 Introduction

SR1's GPIB commands are a collection of text-based commands which can be used to remotely program the SR1 Audio Analyzer. Although nominally called "GPIB" commands, the GPIB command-set can be used over SR1's GPIB (IEEE-488) interface, the RS-232 (serial) interface, or over the Ethernet interface using VXI-11 standard. When used over the IEEE-488 interface SR1 complies with the requirements of IEEE 488.2.

This manual is meant to be a complete reference to the syntax of functionality of SR1's GPIB command set. This manual is **not** intended to be a standalone guide to the features and operation of SR1. When a command is said to query or set a certain feature of the instrument, it is assumed that the reader is familiar with that feature. For a description of SR1's features and operation refer to the **SR1 Operation Manual**. For clarity the command descriptions sometimes contain pictures of portions of SR1 panels to facilitate associating the commands with the portion of the instrument they refer to.

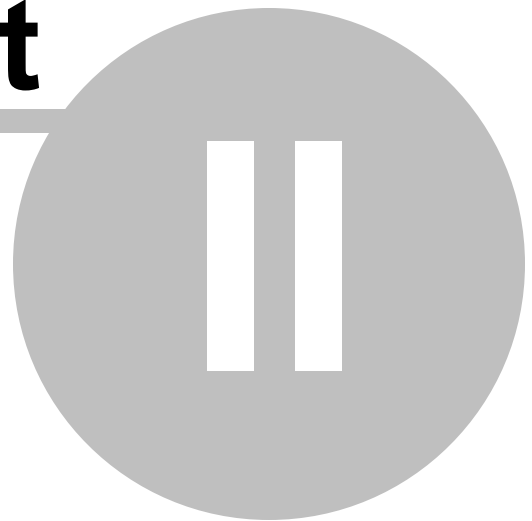
SR1 also can be programmed through the **SR1 Scripting Interface**, based on the Microsoft COM binary interface. The SR1 Scripting commands are used when using the internal scripting interface or when programming the through Visual Basic, Microsoft Office, or other COM enabled applications. The SR1 Scripting interface is documented separately in the SR1 Scripting Manual.

## 1.1 Revision History

Revision	Date	Comments
0.7	7/09	First preliminary version (incomplete).
1.0	9/10	First complete version

***GPIB***

**Part**



## 2 GPIB

### 2.1 Using GPIB Commands

#### **Objects**

Each of SR1's GPIB commands belong to a particular **Object**. Each Object represents a functionally related group of SR1 features. Objects roughly, but not exactly, correspond to the different SR1 panels. The objects are grouped hierarchically: thus the sine waveform object belongs to the object that represents a generator channel, which in turn belongs to the object which represents the generator. The position of any object in the hierarchy is described by an alphanumeric string which starts at the root, and ends at the desired object. For instance, the string which describes a sine waveform on channel A of the analog generator would be:

```
:AnlgGen:Ch(0):Sine(0)
```

Each object in the string is separated by the colon character (:). The initial colon represents the root of the object hierarchy. (In practice, it is not necessary to send the initial colon character but it is included here for completeness.) The first object referenced is the Analog Generator. The second portion of the string specifies the A channel of the analog generator. In general, whenever several lower level objects belong to the same higher level object the specific object to be used in the command is specified by an argument enclosed by parentheses, e.g. (0). The A channel of the analog generator can contain up to 4 waveforms of different types. The final segment of the example string references the first sine waveform belonging to the A channel of the analog generator.

#### **Commands**

Each object can contain several commands. Commands may be simple commands which take no arguments and return no values. For instance:

```
:Displays:Graph(10):Trace(101):AutoScale
```

instructs SR1 to autoscale the trace with trace ID 101 belonging to the graph with ID 10. Note that to send a command it is necessary to send the object specifier string, followed by a colon, followed by the command name.

#### **Command Parameters**

Some commands require the specification of one or more additional parameters. For instance

```
:Sweep:Source(0):IntNumSteps 100
```

instructs SR1 to set the number of steps for sweep source 0 (the inner sweep) to 100. Note that the parameter is separated from the command by a space. In instances where more than parameter is required each additional parameter is separated from the first by a comma.

#### **Parameter Enumerations**

Many command arguments and return values are integers where the actual integer value is not important, but rather each integer value is assigned a specific meaning. These values are called enumerations, and in SR1 these integers are associated with ASCII text mnemonics which indicate the meaning. The most common enumeration, for instance, as both a command argument and return value, is:

```
variable <bool> {False=0|True=1}
```

This indicates that the text string "False" is associated with the integer value 0, and "True" is associated with 1. When sending GPIB commands, enumeration strings or the underlying integer values can be used interchangeably. For instance, it doesn't matter if you send:



```
:SetJitter True
```

or

```
:SetJitter 1
```

For values returned from queries, the "Use Enum" checkbox on the Preferences Panel determines whether SR1 will send the integer or the associated string. So for instance in response to

```
:DigIO:ImpairJitterWaveform?
```

SR1 will respond with "1" if "Use Enum" is off, and "jtSine" if "Use Enum" is on.

### Queries

Commands which return values are referred to as Queries. In general, each command to set a parameter has a corresponding query to read the value of that parameter. The query is formed by adding a "?" to the end of the command name. For example, the query corresponding to the `IntNumSteps` command shown above is:

```
:Sweep:Source(0):IntNumSteps?
```

to which the response of SR1 might be:

```
:Sweep:Source(0):IntNumSteps 100
```

In this manual, SR1 responses will be shown in `bold` type to differentiate them from commands sent to the instrument.

## Command List Conventions

The commands in the following sections are organized hierarchically by object. Each section begins with the object specifier, and if necessary, a description of any arguments necessary to completely specify the object. Next, each command associated with that object is listed. As an example consider this command taken from the Digital I/O object:

### ImpairJitterWaveform

**Command Syntax:** `:DigIO:ImpairJitterWaveform Value [, AllowCoercion=True]`

**Command Argument(s):** `Value <int> {jtOff=0|jtSine=1|jtSquare=2|jtNoise=3|jtNoiseBP=4}`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:DigIO:ImpairJitterWaveform jtSine`

**Related Command(s):** `ImpairJitterWaveform?`

**Description:** Sets the type of Jitter applied to the digital audio output carrier signal.

The top line lists the name of the command. Several commands have alternate long and short forms, and if so, both forms will be listed. The Command Syntax lists the complete command string including any arguments in the object specifier or the command. Items in italics such as *Value* represent the name of the argument, not the literal argument. Arguments enclosed in [square brackets] are optional arguments and are not required to be sent. The default value for optional arguments is shown where appropriate. The Command Arguments list each argument described in the Command Syntax along with the type of argument. The type is shown in <angle brackets>. Integer types, such as <int> and <bool>, may be sent as integer values, or as enumeration strings. When enumerations are allowed, the possible values for the enumerations along with the equivalent integer values are listed in curly brackets. For example the two command strings:

```
:DigIO:ImpairJitterWaveform jtSine, True
```

and

```
:DigIO:ImpairJitterWaveform 1, 1
```

are equivalent.

Below the argument description an example of the command is given with all object and command arguments filled in with typical values. If the command is a query a typical response is given. Note that the examples do not give every possible legal form of the command or response, just one typical case. The Related Commands line lists any related commands. Many commands come in pairs corresponding to the query and non-query form of the command. For instance the `ImpairJitterWaveform` command which sets the jitter type has the related command `ImpairJitterWaveform?` which queries the jitter type. Finally, the description gives a brief description of the function of the command and any additional information about how it might be used in a real application.

### **Allow Coercion**

Many commands to set values include the optional bool argument [`AllowCoercion=True`]. If left at its default value of true, SR1 will coerce out of range arguments to an allowed value. This done by either limiting the supplied value to within the allowed range for that particular parameter, or setting the parameter to some default value if the command is attempting to set it to an illegal value. If "Allow Coercion" is explicitly set to false in a command, SR1 will issue a "Command Error" each time a command attempts to set a parameter outside its allowed range.

### **Units and GPIB Commands**

Many of the quantities queried and set by GPIB commands have units associated with them, and most can be set in more than one unit. Queries that access unit-ed quantities include the optional argument [`ValueUnit`], of type `<unitstring>` allowing specification of the exact units in which the quantity will be returned. For instance, the query `:DigIO:OutputCarrierAmpBal? [ValueUnit]` which returns the digital audio output balanced carrier amplitude can be sent as:

```
DigIO:OutputCarrierAmpBal? Vpp
2.0
```

or:

```
DigIO:OutputCarrierAmpBal? Vp
1.0
```

The [`ValueUnit`] argument is optional. If it is not sent, SR1 will always return the value in whatever units are currently used. (Note that the Preferences Panel include an option which causes SR1 to append unit strings to query results.)

A partial list of units strings is found in the table below. In general, any unit string which is displayed on an SR1 panel will also be recognized by GPIB commands.

<b>Unit String</b>	<b>Units</b>
<b>Amplitude Units</b>	
Vp	Volts peak.
Vpp	Volts peak-to-peak.
Vrms	Vots RMS.
dBV	Decibels relative to 1 Vrms.

dBm	Decibels relative to 1 mW.
dBu	Decibels relative to
dBr	Decibels relative to the dBr reference for object referenced by the command.
dBV <sub>rms</sub>	Decibels relative to 1V <sub>rms</sub> .
W	Watts.
FFS	Fraction of Full Scale.
dBFS	Decibels relative to 1 FFS.
%FS pctFS	Percentage of Full Scale.
<b>Phase Units</b>	
deg	Degrees
rad	Radians
<b>Frequency Units</b>	
Hz	Hertz
F_R	Frequency relative to the frequency reference of the object referenced by the command. For example 2.3 F_R when the reference is 1 kHz implies a frequency of 2.3 kHz.
Octs Octaves	Octaves relative to the frequency reference of the object referenced by the command.
Decs Decades	Decades relative to the frequency reference of the object referenced by the command.
Cts Cents	Cents relative to the frequency reference of the object referenced by the command.
%Ref	Percentage of the frequency reference of the object referenced by the command.
dHz	Frequency Difference (in Hz) from the frequency reference of the object referenced by the command.
%Hz	Percentage of the frequency reference of the object referenced by the command.
ppm	Parts-per-million relative to the frequency reference of the object referenced by the command.
<b>Other Units</b>	
s sec	Seconds.
ohms	Ohms.
CycA CyclesA	Cycles of the Generator A-channel frequency.
CycB CyclesB	Cycles of the Generator B-channel frequency.
UI	Unit intervals
dec	Decimal. (Used for digital generator amplitudes)

hex	Hexadecimal. (Used for digital generator amplitudes)
-----	--

Commands that set the values of unit-ed quantities include a *Value* argument of type <unit>. A <unit> argument consists of a number followed by an optional <unitstring> (see table above). For instance:

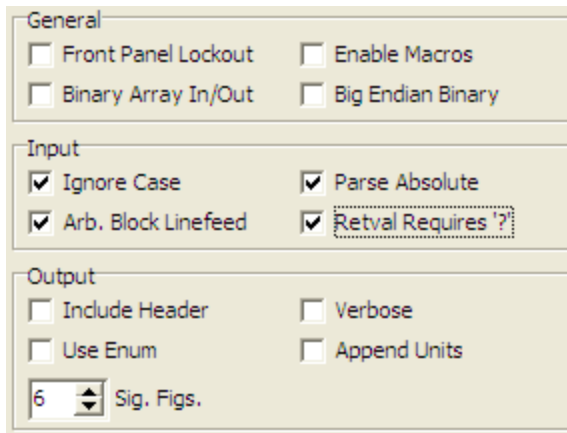
```
DigIO:OutputCarrierAmpBal 2.0 Vpp
```

The <unitstring> is optional. If it is not sent, SR1 will use the current units of the quantity referenced by the command to interpret the argument. The current units for any quantity can be queried by sending the query form of the command followed by the string "unit". SR1 responds with the string indicating the current units. For instance:

```
DigIO:OutputCarrierAmpBal? Unit
Vpp
```

## Command Options

Several options can be set on the SR1 preferences panel which affect how GPIB commands and responses are formatted.



### **Ignore Case (Input)**

The default behavior for SR1, and the behavior specified in the IEEE-488.2 standard, is to ignore the case of commands sent to the instrument. By un-checking "Ignore Case", the user is instructing SR1 to require that all commands be sent exactly as written in the command description. In this case commands that are spelled correctly but have the incorrect case, will trigger a Command Error.

### **Parse Absolute (Input)**

The default behavior for SR1, and the behavior specified in the IEEE-488.2 standard, requires the full object specifier string to be sent with each command. For instance, to set the amplitude, frequency, and turn "on" a sine waveform one would send:

```
:AnlgGen:Ch(A):Sine(0):Freq 1000.0 Hz
:AnlgGen:Ch(A):Sine(0):Amp 3.1 Vrms
:AnlgGen:Ch(A):Sine(0):On True
```

When "Parse Absolute" is unchecked, the object referenced by the first object specifier is assumed to be the target of all subsequent commands until a new object specifier is sent. Thus, the list of commands above could be abbreviated as:

```
:AnlgGen:Ch(A):Sine(0):Freq 1000.0 Hz
Amp 3.1 Vrms
```

On True

#### ***Include Header (Output)***

When "Include Header" is checked SR1 will send the object specifier and command in its response. For example:

```
:AnlgGen:Ch(A):Sine(0):Freq? Hz
:AnlgGen:Ch(A):Sine(0):Freq 1000 Hz
```

Unchecked, SR1 omits the object and command and simply sends the the value:

```
:AnlgGen:Ch(A):Sine(0):Freq? Hz
1000 Hz
```

In the example given in each query command, the response header will be enclosed in [square brackets] to indicate that it may or may not be set depending on the setting of the "Include Header" option.

#### ***Append Units (Output)***

When checked, SR1 will include a string designating the units of the response with the

```
:AnlgGen:Ch(A):Sine(0):Freq? Hz
1000.0 Hz
```

Since the units of the response can be specified in the command, the inclusion of the "Hz" string in the response may be redundant. Unchecking "Append Units" causes SR1 to omit the unit specifier:

```
:AnlgGen:Ch(A):Sine(0):Freq? Hz
1000.0
```

Note that if a unit string is not specified in the original query, SR1 will send the result in the "current" units for that parameter. The current units may be queried by sending the query form of the command followed by the string "unit". SR1 responds with the string indicating the current units. For instance:

```
:AnlgGen:Ch(A):Sine(0):Freq? Unit
Hz
```

#### ***Use Enum (Output)***

Unchecked, enumerated responses will be returned as integers. For instance:

```
:DigIO:ImpairJitterWaveform?
3
```

When checked, SR1 will use enumeration strings in the response:

```
:DigIO:ImpairJitterWaveform?
jtnoise
```

#### ***Verbose (Output)***

When checked, SR1 sends the long form of commands with multiple forms when sending headers in responses. When un-checked, the short form is used.

#### ***Significant Figures***

Sets the default number of significant figures sent with each floating-point response.

## 2.2 GPIB Status Model

SR1 implements a hierarchical status model typical of IEEE-488.2 instruments. Three 32-bit **instrument status registers** (ISRs) indicate the occurrence of different status conditions. Each bit in the ISR can be enabled by setting a bit in a corresponding **enable register** (ISE). The setting of any enabled bit in one of the ISRs causes a summary bit to be set in the **Status Byte** (SB) register. In addition to the SR1-specific ISRs, the IEEE-488.2 standard defines a **Standard Event Status Register** (ESR) which contains bits which reflect various standard status conditions along with its own enable register, and a summary bit in the SB register. Finally, the Status Byte register itself has its own enable register (SRE). The setting of any enabled bits in the SB register causes SR1 to issue a GPIB Service Request (SRQ). The value of the SB register is what is returned when SR1 is serial polled over GPIB.

### Instrument Status Registers

Instrument Status Register 0 (ISR0)		
Bit	Weight	Description
0	1	Analog channel A input scale change.
1	2	Analog channel B input scale change.
2	4	Analog channel A high voltage protection trip.
3	8	Analog channel B high voltage protection trip.
4	16	Sweep started.
5	32	Sweep point begin.
6	64	Sweep point timeout.
7	128	Sweep point done.
8	256	Sweep completed.
9	512	A0 analyzer triggered.
10	1024	A1 analyzer triggered.
11	2048	A new value of watched measurement 0 has been computed. Watched measurements are configured on the "Config" tab of SR1's Events panel
12	4096	A new value of watched measurement 1 has been computed.
13	8192	A new value of watched measurement 2 has been computed.
14	16384	A new value of watched measurement 3 has been computed.
15	32768	A new value of watched measurement 4 has been computed.
16	64536	Digitizer has finished analysis.

Instrument Status Register 1 (ISR1)		
Bit	Weight	Description
0	1	Digital input, channel A validity bit change.
1	2	Digital input, channel B validity bit change.
2	4	Digital input, unlock status change.
3	8	Digital input, biphas error status change.
4	16	Digital input, parity error change.
5	32	Digital input, professional/consumer received channel status format change.
6	64	Digital input, copyright bit change.
7	128	Digital input, emphasis format change.

8	256	Digital input, CRC error status change.
		The next ten status conditions make use of the watched channel status bytes configured on the "Config" tab of SR1's events panel. Each of the 5 watched bytes (WCSB0-WCSB4) can be configured to "watch" one of the received channel status bytes (0-23). The status bit is set when any bit in the watched byte changes in the received channel status for the indicated channel.
9	512	A channel WCSB0 change.
10	1024	B channel WCSB0 change.
11	2048	A channel WCSB1 change.
12	4096	B channel WCSB1 change.
13	8192	A channel WCSB2 change.
14	16384	B channel WCSB2 change.
15	32768	A channel WCSB3 change.
16	65536	B channel WCSB3 change.
17	131072	A channel WCSB4 change.
18	262144	B channel WCSB4 change.
19	524288	A channel user bit activity. (Any change in any of the received user bits)
20	1048576	B channel user bit activity. (Any change in any of the received user bits)

Instrument Status Register 2 (ISR2)		
Bit	Weight	Description
0	1	Keypad press.
1	2	Knob movement.
2	4	Warning has occurred.
3	8	Critical Error has occurred.
4	16	Script timeout.
5	32	Script error.
6	64	User event.
7	128	Bar chart limit exceeded.
8	256	Graph limit exceeded.
9	512	Eye diagram limit exceeded.

## IEEE-488.2 Status Registers

Standard Event Status Register (ESR)			
Bit	Weight	Name	Description
0	1	OPC	Operation complete. Set by the *OPC command and used for synchronizing GPIB communications.
1	2	WAI	unimplemented by SR1
2	4	QYE	Set when an attempt is made to read from the instrument and the output queue is empty, or when an error occurs which force the clearing of the output queue.
3	8	DDE	Device Dependent Error. SR1 implements this error as Value Rejected. This occurs when an out of range parameter is sent and the "Allow Coercion" flag is false.
4	16	EXE	Execution Error. Unable to convert parameters to the required type. Wrong number of

			parameters.
5	32	CME	Command Error. Unable to parse the received command.
6	64	URQ	unimplemented by SR1
7	128	PON	Power On. This bit is set when the SR1 application starts.

<b>The Status Byte Register (SB)</b>			
<b>Bit</b>	<b>Weight</b>	<b>Name</b>	<b>Description</b>
0	1	IS0	Instrument Status 0 summary bit. An unmasked bit in ISR0 has been set.
1	2	IS1	Instrument Status 1 summary bit. An unmasked bit in ISR1 has been set.
2	4	IS2	Instrument Status 2 summary bit. An unmasked bit in ISR2 has been set.
3	8	IS3	Instrument Status 3 summary bit. An unmasked bit in ISR3 has been set.
4	16	MAV	A message is available in SR1's output queue.
5	32	ESB	An unmasked bit in the Standard Event Status Register has been set.
6	64	MSS	Master Status Summary. An unmasked bit in the Status Byte register has been set. When MSS is set, SR1 issues a service request (SRQ).
7	128	IS4	Instrument Status 4 summary bit. An unmasked bit in ISR0 has been set.



## 2.3 GPIB Command Reference

### 2.3.1 GPIB Common Commands

#### Status Commands

##### \*IDN?

**Command Syntax:** \*IDN?

**Response Syntax:** IDstring

**Response Argument(s):** IDString <string>

**Example:** \*IDN?

**Stanford\_Research\_Systems,SR1,s/n104014,ver1.0.12.0**

**Description:** Returns the identification string for SR1. Per IEEE488.2, the identification string consists of the company name, the model name, the serial number, and the version number.

##### \*ESR?

**Command Syntax:** \*ESR?

**Response Syntax:** value

**Response Argument(s):** value<int>

**Example:** \*ESR?

**64**

**Description:** Returns the current value of the Standard Event Status register (ESR). Bits in this register are sticky, i.e. after being set they stay on until cleared by reading the register. The Standard Event Status register contains the following event flags:

Standard Event Status Register (ESR)			
Bit	Weight	Name	Description
0	1	OPC	Operation complete. Set by the *OPC command and used for synchronizing GPIB communications.
1	2	-----	unimplemented by SR1
2	4	QYE	Set when an attempt is made to read from the instrument and the output queue is empty, or when an error occurs which force the clearing of the output queue.
3	8	DD E	Device Dependent Error. SR1 implements this error as Value Rejected. This occurs when an out of range parameter is sent and the "Allow Coercion" flag is false.
4	16	EX E	Execution Error. Unable to convert parameters to the required type. Wrong number of parameters.
5	32	CM E	Command Error. Unable to parse the received command.
6	64	-----	unimplemented by SR1
7	128	PON	Power On. This bit is set when the SR1 application starts.

**\*ESE?***Command Syntax:* \*ESE?*Response Syntax:* value*Response Argument(s):* value<int>*Example:* \*ESE?**64**

*Description:* Returns the value of the Standard Event Enable register. This register is bitwise and-ed with the Standard Event Status register whenever a bit in the status register is set and if the result is non-zero the ESB bit in the Status Byte register is set.

**\*ESE***Command Syntax:* \*ESE value*Command Argument(s):* value<int>*Example:* \*ESE 64

*Description:* Sets the value of the Standard Event Enable register. This register is and-ed with the Standard Event Status register whenever a bit in the status register is set and if the result is non-zero the ESB bit in the Status Byte register is set.

**\*STB?***Command Syntax:* \*STB?*Response Syntax:* value*Response Argument(s):* value<int>*Example:* \*STB?**1**

*Description:* Returns the current value of the Status Byte register (SB). The Status Byte register contains the following event flags:

Bit	Weight	Description
0	1	IS0 (Instrument Status 0)
1	2	IS0 (Instrument Status 1)
2	4	IS0 (Instrument Status 2)
3	8	IS0 (Instrument Status 3)
4	16	MAV (Message Available)
5	32	ESB (Event Status Summary)
6	64	MSS (Master Status Summary)
7	128	IS4 (Instrument Status 4)

## \*SRE?

*Command Syntax:* \*SRE?

*Response Syntax:* value

*Response Argument(s):* value<int>

*Example:* \*SRE?

64

*Description:* Returns the current value of the Service Request Enable register. When a bit in the Status Byte register is set the Status Byte is and-ed with the contents of the Service Request Enable register. If the result is non-zero the MSS bit in the Status Byte is set and a Service Request is issued.

## \*SRE

*Command Syntax:* \*ESE value

*Command Argument(s):* value<int>

*Example:* \*SRE 64

*Description:* Sets the value of the Service Request Enable register. When a bit in the Status Byte register is set the Status Byte is and-ed with the contents of the Service Request Enable register. If the result is non-zero the MSS bit in the Status Byte is set and a Service Request is issued.

## IESR?

*Command Syntax:* \*IESR (regIndex)?

*Command Argument(s):* regIndex <int (0-4)>

*Response Syntax:* value

*Response Argument(s):* value<int>

*Example:* IESR (2) ?

65

*Description:* Queries the value of the specified instrument status register.

## INSE?

*Command Syntax:* \*IESE( regIndex)?

*Command Argument(s):* regIndex <int (0-4)>

*Response Syntax:* value

*Response Argument(s):* value <int>

*Example:* IESE (2) ?

1

*Description:* Queries the value of the specified instrument status enable register. When a bit is set in the corresponding instrument status register, the status register is and-ed with the enable register and if the result is non-zero the summary bit is set in the Status Byte register.

## IESE

**Command Syntax:** IESE (*regIndex*) *value*

**Command Argument(s):** *regIndex* <int (0-4)>  
*value* <int>

**Example:** IESE (2) 65

**Description:** Sets the value of the specified instrument status enable register. When a bit is set in the corresponding instrument status register, the status register is and-ed with the enable register and if the result is non-zero the summary bit is set in the Status Byte register.

## \*CLS

**Command Syntax:** \*CLS

**Example:** \*CLS

**Description:** Clears the Standard Event Status Register (ESR) and all SR1 Instrument Status Registers.

## General Purpose Commands

### \*OPC

**Command Syntax:** \*OPC

**Example:** \*OPC

**Description:** Sets the OPC bit in the Standard Event Status Register. If this bit is configured to cause a Service Request, and the \*OPC command is sent at the end of a string of commands, the occurrence of the Service Request can be used as an indicator that SR1 has finished processing the string of commands.

### \*OPC?

**Command Syntax:** \*OPC?

**Response Syntax:** *value* <int>

**Example:** \*OPC?

1

**Description:** Always returns the value "1" but does not affect the Standard Event Status Register. Receipt of the "1" sent by this command can be used as a marker that SR1 has finished processing all previous commands.

### \*WAI

**Command Syntax:** \*WAI

**Example:** \*WAI

**Description:** \*WAI returns no error, but is not currently implemented by SR1.

## \*TST

**Command Syntax:** \*TST

**Response Syntax:** *value* <bool> {False=0 | True=1}

**Example:** \*TST  
1

**Description:** Returns a bool indicating the current status of the SR1 hardware. A "True" returned indicates that all the items on the Hardware Status panel are "green." A false indicates that one or more hardware problems have been detected.

## \*RST

**Command Syntax:** \*RST

**Example:** \*RST

**Description:** \*RST loads a configuration file corresponding to the default configuration of SR1.

## \*TRG

**Command Syntax:** \*TRG

**Example:** \*TRG

**Description:** Triggers both the A0 and A1 analyzers.

## \*SAV

**Command Syntax:** \*SAV *savIndex*

**Command Argument(s):** *savIndex* <int 1-9>

**Example:** \*SAV 2

**Description:** Saves the entire instrument configuration to one of the numbered slots. Slots 1-9 may be saved to, slot 0 is reserved for the default instrument configuration and is read-only.

## \*RCL

**Command Syntax:** \*RCL *rclIndex*

**Command Argument(s):** *rclIndex*<int 0-9>

**Example:** \*RCL 1

**Description:** Recalls the instrument configurations in the specified slot. Slot 0 is reserved for the default SR1 configuration.

## Macro Commands

### **\*EMC**

*Command Syntax:* \*EMC *value*

*Command Argument(s):* *value*<bool> {False=0 | True=1}

*Example:* \*EMC False

*Description:* Enables (True) and disables (false) expansion of macro commands. A macro command sent while macros are disabled results in a Execution Error.

### **\*EMC?**

*Command Syntax:* \*EMC?

*Response Syntax:* *value*<bool> {False=0 | True=1}

*Example:* \*EMC?

**False**

*Description:* Queries whether macros are enabled.

## \*DMC

**Command Syntax:** \*DMC label , macro

**Command Argument(s):** label <string>  
macro <arbitrary block program data>

**Example:** \*DMC "SETGEN", #276:AnlgGen:Ch(A):Sine(0):Freq 1000Hz;  
AnlgGen:Ch(A):Sine(0):Amp 1Vrms

**Description:** Define Macro Command. The label argument specifies the name of the macro which can be any string which does not begin with the character \* (so as not to interfere with GPIB common commands). The macro argument gives the actual commands which the macro will be expanded into. In the example given, the macro named "SETGEN" will set the sine frequency to 1 kHz and the amplitude to 1 Vrms. Parameters can be passed to macros during expansion. The placeholder for the first passed parameter is definition string by \$1, the second by \$2, etc. Thus, we could rewrite the example macro definition using replaceable parameters as follows:  
\*DMC "SETGEN" #277:AnlgGen:Ch(A):Sine(0):Freq \$1 Hz;  
AnlgGen:Ch(A):Sine(0):Amp \$2 Vrms  
and the macro could be invoked as follows:  
SETGEN 1000,1.0

The "#277" which starts the macro definitions specifies the length of the definition that follows. The first character after the "#" gives the number, N, of decimal digits in the length specifier. The next N decimal digits actually specify the length of the macro definition, including spaces, that follows. Thus "#277" means that the number of digits in the length of the macro definition is 2, and that the length of the macro definition is 77 characters. To avoid counting characters it is possible to use the arbitrary length arbitrary block data format by preceding the macro definition with #0, e.g.:

```
*DMC "SETGEN", #0AnlgGen:Ch(A):Sine(0):Freq $1 Hz;  
AnlgGen:Ch(A):Sine(0):Amp $2 Vrms
```

When using the #0 be sure to terminate the macro definition with a linefeed (0x0a) character, or to turn off the "Arb. Block Linefeed" option on the Remote tab of SRI's preference panel.

## \*GMC?

**Command Syntax:** \*GMC macroname

**Command Argument(s):** macroname<string>

**Response Syntax:** [macrodef]

**Response Argument:** macrodef<arbitrary block program data>

**Example:** \*GMC? SETGEN  
#276:AnlgGen:Ch(A):Sine(0):Freq 1000Hz;AnlgGen:Ch(A):  
Sine(0):Amp 1Vrms

**Description:** Get Macro Command. Returns the macro definition for the specified macro. If no macro with that name is currently defined an Execution Error results.

## \*LMC?

**Command Syntax:** \*LMC

**Response Syntax:** [macroname1, macroname2, ...]

**Response Arguments:** macronamei<string>

**Example:** \*LMC

"SETGEN", "SETALYZER", "SETDIGIO"

**Description:** List Macro Command. This command returns a list of the names of all currently defined macros.

## \*PMC

**Command Syntax:** \*PMC

**Example:** \*PMC

**Description:** Purge all Macros. This command deletes the definitions of any currently defined macros.

## \*RMC

**Command Syntax:** \*RMC macroname

**Command Argument(s):** macroname<string>

**Example:** \*RMC SETGEN

**Description:** Remove Macro Command. Removes the macro definition for the macro specified in the argument. If no macro with that name is currently defined an Execution Error results.

## \*DDT

**Command Syntax:** \*DDT macro

**Command Argument(s):** macro <arbitrary block program data>

**Example:** \*DDT "SETGEN", #0:AnlgGen:Ch(A):Sine(0):Freq 1000Hz;  
AnlgGen:Ch(A):Sine(0):Amp 1Vrms

**Description:** Define Macro Trigger Command. Defines a special macro which is executed each time a Group Execute Trigger (GET) GPIB message is received or the \*TRG common command is received.



## 2.3.2 Form Commands

Many objects are associated with a particular form and share a common set of commands for opening, closing, and manipulating their forms. Some of these commands return or take as arguments a FormID. The FormID is an integer which provides a handle to a particular instance of a form. In general, commands that take a FormID refer only to that particular instance of the form while commands that do not use a FormID refer to all instances of that form on all pages of the page control.

### OpenForm

*Command Syntax:* :ObjString:OpenForm

*Command Argument(s):* None

*Example:* :ObjString:OpenForm

*Description:* Opens a form on the current page of the page control.

### OpenFormwID?

*Command Syntax:* :ObjString:OpenFormwID?

*Command Argument(s):* None

*Response Syntax:* [:ObjString:OpenFormwID] FormID

*Response Argument(s):* FormID <int>

*Example:* :ObjString:OpenFormwID?

**[:ObjString:OpenFormwID] 101**

*Description:* Opens a form on the current page of the page control and returns its FormID.

### CloseForm

*Command Syntax:* :ObjString:CloseForm FormID

*Command Argument(s):* FormID <int>

*Example:* :ObjString:CloseForm 23

*Description:* Closes the particular instance of the form with the given FormID.

### CloseForms

*Command Syntax:* :ObjString:CloseForms

*Command Argument(s):* None

*Example:* :ObjString:CloseForms

*Description:* Closes all instances of the form on all pages of the page control.

## FormCount?

*Command Syntax:* :ObjString:FormCount?

*Command Argument(s):* None

*Response Syntax:* [:ObjString:FormCount] Count

*Response Argument(s):* Count <int>

*Example:* :ObjString:FormCount?  
[:ObjString:FormCount] 3

*Description:* Returns the number of open forms on all pages of the page control corresponding to ObjString.

## FormID?

*Command Syntax:* :ObjString:FormID? Index

*Command Argument(s):* Index <int>

*Response Syntax:* [:Alyz(i):Jitter:FormID] FormID

*Response Argument(s):* FormID <int>

*Example:* :ObjString:FormID? Value  
[:ObjString:FormID] 101

*Description:* Returns the FormID of the Index<sup>th</sup> form of the given type. Index=0 corresponds to the first form.

### 2.3.3 Analog Inputs

<b>Object:</b>	<b>:AnlgInputs :AnlgIn</b>
<b>Object Argument(s):</b>	None
<b>Description:</b>	Properties of the Analog Inputs not specific to a particular input channel.

#### HiResSampleRate?

**Command Syntax:** :AnlgInputs:HiResSampleRate?

**Command Argument(s):** None

**Response Syntax:** [:AnlgInputs:HiResSampleRate ]Value

**Response Argument(s):** Value <int> {srHz64k=0 | srHz128k=1 | srOSR=2 | srOSRx2=3}

**Example:** :AnlgInputs:HiResSampleRate?

**[ :AnlgInputs:HiResSampleRate ] srHz64k**

**Related Command(s):** HiResSampleRate

**Description:** Queries the sample rate enumeration of the Hi-Resolution converter.

#### HiResSampleRate

**Command Syntax:** :AnlgInputs:HiResSampleRate Value [, AllowCoercion]

**Command Argument(s):** Value <int> {srHz64k=0 | srHz128k=1 | srOSR=2 | srOSRx2=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgInputs:HiResSampleRate srHz64k

**Related Command(s):** HiResSampleRate?

**Description:** Sets the Hi-Resolution converter sample rate.

#### Form Commands Supported :

**:AnlgIn:OpenForm**

**:AnlgIn:OpenFormwID?**

**:AnlgIn:CloseForm**

**:AnlgIn:CloseForms**

**:AnlgIn:FormCount?**

**:AnlgIn:FormID?**

### 2.3.4 Analog Input Channel

<b>Object:</b>	<b>:Ch(<i>Ch</i>)</b>
<b>Object Argument(s):</b>	<i>ch</i> <char> {A   B} or <i>ch</i> <int> {0 1}
<b>Description:</b>	Properties of the Analog Inputs specific to a particular input channel.

#### AutoRange?

**Command Syntax:** :Ch(*Ch*):AutoRange?

**Command Argument(s):** None

**Response Syntax:** [:Ch(*Ch*):AutoRange ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Ch(0):AutoRange?  
[:Ch(0):AutoRange ]True

**Related Command(s):** AutoRange

**Description:** Queries the AutoRange status for the selected input.

#### AutoRange

**Command Syntax:** :Ch(*Ch*):AutoRange *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(A):AutoRange True

**Related Command(s):** AutoRange?

**Description:** Sets the AutoRange status for the selected input.

#### Coupling?

**Command Syntax:** :Ch(*Ch*):Coupling?

**Response Syntax:** [:Ch(*Ch*):Coupling ]*Value*

**Response Argument(s):** *Value* <int> {cpIAC=0 | cpIDC=1}

**Example:** :Ch(1):Coupling?  
[:Ch(1):Coupling ] cpIAC

**Related Command(s):** Coupling

**Description:** Queries the coupling selection for the selected input channel.

## Coupling

**Command Syntax:** :Ch(*Ch*):Coupling *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cplAC=0 | cplDC=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(B):Coupling *Value* [, *AllowCoercion*]

**Related Command(s):** Coupling?

**Description:** Sets the coupling selection for the selected input channel.

## OptionalFilter?

**Command Syntax:** :Ch(*Ch*):OptionalFilter?

**Response Syntax:** [:Ch(*Ch*):OptionalFilter ]*Value*

**Response Argument(s):** *Value* <int> {ifNone=0 | ifFilter1=1 | ifFilter2=2 | ifFilter3=3 | ifFilter4=4 | ifGround=5}

**Example:** :Ch(A):OptionalFilter?  
 [:Ch(A):OptionalFilter ]ifNone

**Related Command(s):** OptionalFilter

**Description:** Queries the Optional filter currently inserted in the selected channel signal path.

## OptionalFilter

**Command Syntax:** :Ch(*Ch*):OptionalFilter *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ifNone=0 | ifFilter1=1 | ifFilter2=2 | ifFilter3=3 | ifFilter4=4 | ifGround=5}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(B):OptionalFilter ifFilter2

**Related Command(s):** OptionalFilter?

**Description:** Sets the Optional filter currently inserted into the selected channel signal path.

## Range?

**Command Syntax:** :Ch(*Ch*):Range? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Ch(*Ch*):Range ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Ch(A):Range? Vrms  
 [:Ch(A):Range ] 16.00 Vrms

**Related Command(s):** Range

**Description:** Queries the input range for the selected input channel.

## Range

**Command Syntax:** :Ch(*Ch*):Range *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(A):Range 16 Vrms

**Related Command(s):** Range?

**Description:** Sets the input range for the selected input channel.

## Source?

**Command Syntax:** :Ch(*Ch*):Source?

**Response Syntax:** [:Ch(*Ch*):Source ]*Value*

**Response Argument(s):** *Value* <int> {aiXLR=0 | aiBNC=1 | aiGenMon=2 | aiDigCommonMode=3}

**Example:** :Ch(B):Source?  
 [:Ch(B):Source ] aiBNC

**Related Command(s):** Source

**Description:** Queries the input source selection for the selected input channel.

## Source

**Command Syntax:** :Ch(*Ch*):Source *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {aiXLR=0 | aiBNC=1 | aiGenMon=2 | aiDigCommonMode=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(0):Source aiGenMon

**Related Command(s):** Source?

**Description:** Sets the input source for the selected input channel.

## Status?

**Command Syntax:** :Ch(*Ch*):Status?

**Response Syntax:** [:Ch(*Ch*):Status ]*Value*

**Response Argument(s):** *Value* <int> {aiUnderRange=0 | aiInRange=1 | aiOverRange=2}

**Example:** :Ch(0):Status?  
 [:Ch(0):Status ] aiUnderRange

**Related Command(s):** Status

**Description:** Queries the range status for the selected input channel.

## Zin?

**Command Syntax:** :Ch(*Ch*):Zin?

**Response Syntax:** [:Ch(*Ch*):Zin ]*Value*

**Response Argument(s):** *Value* <int> {aiHiZ=0 | aiz300=1 | aiz600=2}

**Example:** :Ch(A):Zin?

[ :Ch(A):Zin ] aiz300

**Related Command(s):** Zin

**Description:** Queries the input impedance selection for the selected channel.

## Zin

**Command Syntax:** :Ch(*Ch*):Zin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {aiHiZ=0 | aiz300=1 | aiz600=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Ch(1):Zin 2

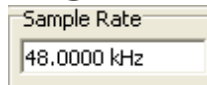
**Related Command(s):** Zin?

**Description:** Sets the input impedance selection for the selected input channel.

### 2.3.5 Digital I/O

<b>Object:</b>	<b>:DigIO</b>
<b>Object Argument(s):</b>	None
<b>Description:</b>	Commands related to features contained on the Digital I/O panel.

## Digital Audio Output Commands



### OSR?

**Command Syntax:** :DigIO:OSR? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:OSR ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:OSR? Hz  
[:DigIO:OSR ]48000

**Related Command(s):** OSR

**Description:** Queries the digital audio output sampling rate.

### OSR

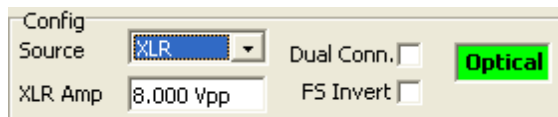
**Command Syntax:** :DigIO:OSR *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:OSR 48000

**Related Command(s):** OSR?

**Description:** Sets the digital audio output sampling rate.



### OutputCarrierAmpBal?

**Command Syntax:** :DigIO:OutputCarrierAmpBal? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:OutputCarrierAmpBal ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:OutputCarrierAmpBal? Vpp  
[:DigIO:OutputCarrierAmpBal ] 10 Vpp

**Related Command(s):** OutputCarrierAmpBal

**Description:** Queries the amplitude of the balanced digital audio carrier signal.



## OutputCarrierAmpBal

**Command Syntax:** :DigIO:OutputCarrierAmpBal *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:OutputCarrierAmpBal 10.0 Vpp

**Related Command(s):** OutputCarrierAmpBal?

**Description:** Sets the amplitude of the balanced digital audio carrer signal.

## OutputCarrierAmpUnbal?

**Command Syntax:** :DigIO:OutputCarrierAmpUnbal? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitsstring>

**Response Syntax:** [:DigIO:OutputCarrierAmpUnbal ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:OutputCarrierAmpUnbal? Vpp  
[:DigIO:OutputCarrierAmpUnbal ] 2.0

**Related Command(s):** OutputCarrierAmpUnbal

**Description:** Queries the amplitude of the unbalanced digital audio carrier signal.

## OutputCarrierAmpUnbal

**Command Syntax:** :DigIO:OutputCarrierAmpUnbal *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:OutputCarrierAmpUnbal 2.0

**Related Command(s):** OutputCarrierAmpUnbal?

**Description:** Sets the amplitude of the unbalanced digital audio carrier signal.

## OutputDualConnector?

**Command Syntax:** :DigIO:OutputDualConnector?

**Response Syntax:** [:DigIO:OutputDualConnector ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigIO:OutputDualConnector?  
[:DigIO:OutputDualConnector ]*Value*

**Related Command(s):** OutputDualConnector

**Description:** Queries the on/off status of the digital audio output dual-connector setting.

## OutputDualConnector

**Command Syntax:** :DigIO:OutputDualConnector *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:OutputDualConnector False

**Related Command(s):** OutputDualConnector?

**Description:** Sets the on/off status of the digital audio output dual-connector setting.

## OpticalOutputActive? OpticalOutActive?

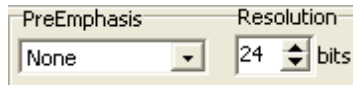
**Command Syntax:** :DigIO:OpticalOutputActive?

**Response Syntax:** [:DigIO:OpticalOutputActive ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigIO:OpticalOutputActive?  
[:DigIO:OpticalOutputActive ]True

**Description:** Queries the active status of the digital audio optical output.



## OutputNumBits?

**Command Syntax:** :DigIO:OutputNumBits?

**Response Syntax:** [:DigIO:OutputNumBits ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:OutputNumBits?  
[:DigIO:OutputNumBits ]24

**Related Command(s):** OutputNumBits

**Description:** Queries the number of bits of output resolution for the digital audio output.

## OutputNumBits

**Command Syntax:** :DigIO:OutputNumBits *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:OutputNumBits *Value* [, *AllowCoercion*]

**Related Command(s):** OutputNumBits?

**Description:** Sets the number of bits of output resolution for the digital audio output.

## OutputSource?

**Command Syntax:** :DigIO:OutputSource?

**Response Syntax:** [:DigIO:OutputSource ]*Value*

**Response Argument(s):** *Value* <int> {dosXLR=0 | dosBNC=1}

**Example:** :DigIO:OutputSource?  
[:DigIO:OutputSource ]dosXLR

**Related Command(s):** OutputSource

**Description:** Queries the connector for the digital audio output.

## OutputSource

**Command Syntax:** `:DigIO:OutputSource Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {dosXLR=0 | dosBNC=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigIO:OutputSource dosBNC, True`

**Related Command(s):** OutputSource?

**Description:** Sets the connector for the digital audio output.

## DigPreemphasis?

**Command Syntax:** `:DigIO:DigPreemphasis?`

**Response Syntax:** `[:DigIO:DigPreemphasis ]Value`

**Response Argument(s):** `Value <int> {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}`

**Example:** `:DigIO:DigPreemphasis?`  
`[:DigIO:DigPreemphasis ]dioNoDeem`

**Related Command(s):** DigPreemphasis

**Description:** Queries the Preemphasis mode of the digital audio output.

## DigPreemphasis

**Command Syntax:** `:DigIO:DigPreemphasis Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigIO:DigPreemphasis dioNoDeem`

**Related Command(s):** DigPreemphasis?

**Description:** Sets the Preemphasis mode of the digital audio output.

## FrameSyncOutInvert?

**Command Syntax:** `:DigIO:FrameSyncOutInvert?`

**Response Syntax:** `[:DigIO:FrameSyncOutInvert ]Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigIO:FrameSyncOutInvert?`  
`[:DigIO:FrameSyncOutInvert ]False`

**Related Command(s):** FrameSyncOutInvert

**Description:** Queries the value of the Frame Sync Invert on the Digital Audio Output tab.

## FrameSyncOutInvert

**Command Syntax:** `:DigIO:FrameSyncOutInvert Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigIO:FrameSyncOutInvert False`

**Related Command(s):** FrameSyncOutInvert?

**Description:** Sets the value of the Frame Sync Invert on the Digital Audio Output tab.

## Digital Audio Input Commands

### ChStatFsARdg?

**Command Syntax:** :DigIO:ChStatFsARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ChStatFsARdg ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ChStatFsARdg? Hz  
[:DigIO:ChStatFsARdg ]48000

**Description:** Query returns the Sampling Rate indicated by the Channel A digital audio status bits.

### ChStatFsBRdg?

**Command Syntax:** :DigIO:ChStatFsBRdg? [*ValueUnit*]

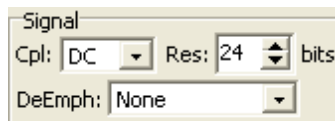
**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ChStatFsBRdg ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ChStatFsBRdg? Hz  
[:DigIO:ChStatFsBRdg ]48000

**Description:** Query returns the Sampling Rate indicated by the Channel B digital audio status bits.



### Deemphasis?

**Command Syntax:** :DigIO:Deemphasis?

**Response Syntax:** [:DigIO:Deemphasis ]*Value*

**Response Argument(s):** *Value* <int> {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}

**Example:** :DigIO:Deemphasis?  
[:DigIO:Deemphasis ]dioNoDeem

**Related Command(s):** Deemphasis

**Description:** Queries the Deemphasis applied to the received digital audio signal.

## Deemphasis

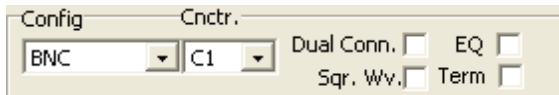
**Command Syntax:** :DigIO:Deemphasis *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:Deemphasis dioNoDeem

**Related Command(s):** Deemphasis?

**Description:** Sets the De-emphasis applied to the received digital audio signal. (Currently only dioNoDeem is supported).



## InputBlockDC?

**Command Syntax:** :DigIO:InputBlockDC?

**Response Syntax:** [:DigIO:InputBlockDC ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigIO:InputBlockDC?

**[:DigIO:InputBlockDC ]False**

**Related Command(s):** InputBlockDC

**Description:** Queries the on/off status of the digital audio input DC blocker.

## InputBlockDC

**Command Syntax:** :DigIO:InputBlockDC *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:InputBlockDC True

**Related Command(s):** InputBlockDC?

**Description:** Sets the on/off status of the digital audio input DC blocker.

## InputCarrierEq?

**Command Syntax:** :DigIO:InputCarrierEq?

**Response Syntax:** [:DigIO:InputCarrierEq ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigIO:InputCarrierEq?

**[:DigIO:InputCarrierEq ]False**

**Related Command(s):** InputCarrierEq

**Description:** Queries the status of the digital audio input EQ.

## InputCarrierEq

**Command Syntax:** :DigIO:InputCarrierEq Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputCarrierEq True

**Related Command(s):** InputCarrierEq?

**Description:** Sets the status of the digital audio input EQ.

## InputConnectorSelect?

**Command Syntax:** :DigIO:InputConnectorSelect?

**Response Syntax:** [:DigIO:InputConnectorSelect ]Value

**Response Argument(s):** Value <int> {dioC0=0 | dioC1=1}

**Example:** :DigIO:InputConnectorSelect?

**[:DigIO:InputConnectorSelect ]dioC0**

**Related Command(s):** InputConnectorSelect

**Description:** Queries the digital audio input connector selection.

## InputConnectorSelect

**Command Syntax:** :DigIO:InputConnectorSelect Value [, AllowCoercion]

**Command Argument(s):** Value <int> {dioC0=0 | dioC1=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputConnectorSelect dioC0

**Related Command(s):** InputConnectorSelect?

**Description:** Sets the digital audio input connector selection.

## InputDualConnector?

**Command Syntax:** :DigIO:InputDualConnector?

**Response Syntax:** [:DigIO:InputDualConnector ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:InputDualConnector?

**[:DigIO:InputDualConnector ]False**

**Related Command(s):** InputDualConnector

**Description:** Queries the status of the digital audio input dual-connector selection.

## InputDualConnector

**Command Syntax:** :DigIO:InputDualConnector Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputDualConnector True

**Related Command(s):** InputDualConnector?

**Description:** Sets the status of the digital audio input dual-connector selection.

## InputNumBits?

**Command Syntax:** :DigIO:InputNumBits?

**Response Syntax:** [:DigIO:InputNumBits ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:InputNumBits?  
[:DigIO:InputNumBits ]24

**Related Command(s):** InputNumBits

**Description:** Queries the resolution (number of bits) of the digital audio input.

## InputNumBits

**Command Syntax:** :DigIO:InputNumBits Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputNumBits 24

**Related Command(s):** InputNumBits?

**Description:** Sets the resolution (number of bits) of the digital audio input.

## InputSource?

**Command Syntax:** :DigIO:InputSource?

**Response Syntax:** [:DigIO:InputSource ]Value

**Response Argument(s):** Value <int> {diXLR=0 | diBNC=1 | diOptical=2 | diGenMon=3 | diNone=4}

**Example:** :DigIO:InputSource?  
[:DigIO:InputSource ]diXLR

**Related Command(s):** InputSource

**Description:** Queries the digital audio input source selection (XLR, BNC, Optical, or GenMon).

## InputSource

**Command Syntax:** :DigIO:InputSource Value [, AllowCoercion]

**Command Argument(s):** Value <int> {diXLR=0 | diBNC=1 | diOptical=2 | diGenMon=3 | diNone=4}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputSource Value diXLR

**Related Command(s):** InputSource?

**Description:** Sets the digital audio input source selection (XLR, BNC, Optical, or GenMon).

## InputTerm?

**Command Syntax:** :DigIO:InputTerm?

**Response Syntax:** [:DigIO:InputTerm ]Value

**Response Argument(s):** Value <int> {dinHiZ=0 | dinLoZ=1}

**Example:** :DigIO:InputTerm?  
[:DigIO:InputTerm ]True

**Related Command(s):** InputTerm

**Description:** Queries the digital audio input termination status.

## InputTerm

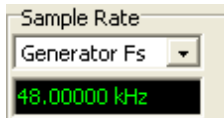
**Command Syntax:** :DigIO:InputTerm Value [, AllowCoercion]

**Command Argument(s):** Value <int> {dinHiZ=0 | dinLoZ=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:InputTerm True

**Related Command(s):** InputTerm?

**Description:** Sets the digital audio input termination status.



## ISRRdg?

**Command Syntax:** :DigIO:ISRRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:DigIO:ISRRdg ]Value

**Response Argument(s):** Value <unit>

**Example:** :DigIO:ISRRdg? Hz  
[:DigIO:ISRRdg 48000

**Description:** Queries the current effective digital audio input sampling rate (ISR). This value reflects the ISR Reference mode selection made with the ISRRef command.

## ISRRef?

**Command Syntax:** :DigIO:ISRRef?

**Response Syntax:** [:DigIO:ISRRef ]Value

**Response Argument(s):** Value <int> {dioGenerator=0 | dioMeasured=1 | dioStatusBits=2 | dioUser=3}

**Example:** :DigIO:ISRRef?  
[:DigIO:ISRRef ]dioGenerator

**Related Command(s):** ISRRef

**Description:** Queries the digital audio input sampling rate mode selection.

## ISRRef

**Command Syntax:** :DigIO:ISRRef Value [, AllowCoercion]

**Command Argument(s):** Value <int> {dioGenerator=0 | dioMeasured=1 | dioStatusBits=2 | dioUser=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ISRRef dioGenerator

**Related Command(s):** ISRRef?

**Description:** Sets the digital audio input sampling rate mode selection.



## UserISRRef?

**Command Syntax:** :DigIO:UserISRRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:UserISRRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:UserISRRef? Hz  
[:DigIO:UserISRRef] 44100

**Related Command(s):** UserISRRef

**Description:** Queries the input sampling frequency used when the input sampling rate mode selection is set to dioUser.

## UserISRRef

**Command Syntax:** :DigIO:UserISRRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:UserISRRef 44100

**Related Command(s):** UserISRRef?

**Description:** Sets the digital audio input sampling rate used when the input sampling rate mode selection is set to dioUser.

## ReceiverStatusRdg? RcvrStatRdg?

**Command Syntax:** :DigIO:ReceiverStatusRdg?

**Response Syntax:** [:DigIO:ReceiverStatusRdg] *Value*

**Response Argument(s):** *Value* <int> {rsmNoErr=0 | rsmParityErr=1 | rsmBiPhaseErr=2 | rsmConfErr=4 | rsmValidityErr=8 | rsmLockErr=16 | rsmCRCErr=32}

**Example:** :DigIO:ReceiverStatusRdg?  
[:DigIO:ReceiverStatusRdg] 7

**Description:** Queries the current status of the digital audio input. Each of the conditions indicated at the bottom of the Digital I/O panel is assigned a value (see the enumerations above) and the returned value is equal to the sum of the values for each active condition.

## ReferenceStatusRdg? RefStatRdg?

**Command Syntax:** :DigIO:ReferenceStatusRdg?

**Response Syntax:** [:DigIO:ReferenceStatusRdg] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ReferenceStatusRdg?  
[:DigIO:ReferenceStatusRdg] 7

**Description:** Queries the current status of the rear-panel AES reference input. Each of the conditions indicated at the bottom of the Digital I/O panel is assigned a value (see the enumerations above) and the returned value is equal to the sum of the values for each active condition.

## SqWaveInput?

**Command Syntax:** :DigIO:SqWaveInput?

**Response Syntax:** [:DigIO:SqWaveInput ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:SqWaveInput?  
[:DigIO:SqWaveInput ]False

**Related Command(s):** SqWaveInput

**Description:** Queries the current value of the digital audio input square wave selection. Used by the Jitter analyzer to determine whether the input is a clock signal (square wave) or a consumer/professional digital audio signal.

## SqWaveInput

**Command Syntax:** :DigIO:SqWaveInput Value [, AllowCoercion]

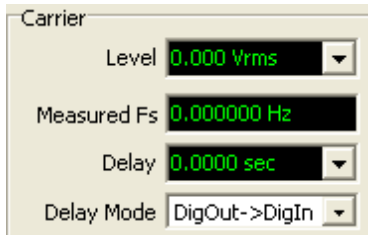
**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:SqWaveInput False

**Related Command(s):** SqWaveInput?

**Description:** Sets the current value of the digital audio input square wave selection. Used by the Jitter analyzer to determine whether the input is a clock signal (square wave) or a consumer/professional digital audio signal.

## *Carrier Status Commands*



### **InputCarrierAmpRdg?**

*Command Syntax:* :DigIO:InputCarrierAmpRdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:DigIO:InputCarrierAmpRdg ]*Value*

*Response Argument(s):* *Value* <unit>

*Example:* :DigIO:InputCarrierAmpRdg? Vpp  
[:DigIO:InputCarrierAmpRdg ]6.88

*Description:* Queries the current amplitude measurement of the digital audio input carrier signal.

### **MeasISRRdg?**

*Command Syntax:* :DigIO:MeasISRRdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:DigIO:MeasISRRdg ]*Value*

*Response Argument(s):* *Value* <unit>

*Example:* :DigIO:MeasISRRdg? [ValueUnit]  
[:DigIO:MeasISRRdg ]48000

*Description:* Queries the *measured* value of the digital audio input sampling rate. If the input is set to dual connector this value will reflect the *physical* sampling rate on each wire rather than the combined logical sampling rate.

### **DelayFromOutRdg?**

*Command Syntax:* :DigIO:DelayFromOutRdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:DigIO:DelayFromOutRdg ]*Value*

*Response Argument(s):* *Value* <unit>

*Example:* :DigIO:DelayFromOutRdg? sec  
[:DigIO:DelayFromOutRdg ]0.00000012

*Description:* Queries the delay value associated with the current "DelayMode".

## DelayMode?

**Command Syntax:** :DigIO:DelayMode?

**Response Syntax:** [:DigIO:DelayMode ]Value

**Response Argument(s):** Value <int> {dioDigOut2DigIn=0 | dioRefOut2DigIn=1}

**Example:** :DigIO:DelayMode?  
[:DigIO:DelayMode ]0

**Related Command(s):** DelayMode

**Description:** Queries the value of the Carrier Status "Delay Mode."

## DelayMode

**Command Syntax:** :DigIO:DelayMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {dioDigOut2DigIn=0 | dioRefOut2DigIn=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:DelayMode 0

**Related Command(s):** DelayMode?

**Description:** Sets the value of the Carrier Status "Delay Mode."

## HiliteChStatusDiff?

**Command Syntax:** :DigIO:HiliteChStatusDiff?

**Response Syntax:** [:DigIO:HiliteChStatusDiff ]Value

**Response Argument(s):** Value <int> {hdNone=0 | hdDiffFromOutput=1 | hdDiffFromOtherCh=2 |  
hdReservedInUse=3}

**Example:** :DigIO:HiliteChStatusDiff?  
[:DigIO:HiliteChStatusDiff ]hdNone

**Related Command(s):** HiliteChStatusDiff

**Description:** Queries the highlight mode used by the channel status and user status forms.

## HiliteChStatusDiff

**Command Syntax:** :DigIO:HiliteChStatusDiff Value [, AllowCoercion]

**Command Argument(s):** Value <int> {hdNone=0 | hdDiffFromOutput=1 | hdDiffFromOtherCh=2 |  
hdReservedInUse=3}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:HiliteChStatusDiff hdDiffFromOutput

**Related Command(s):** HiliteChStatusDiff?

**Description:** Sets the highlight mode used by the channel status and user forms.

## Data/Active Bits Indicator

### ActiveBitsA?

**Command Syntax:** :DigIO:ActiveBitsA?

**Response Syntax:** [:DigIO:ActiveBitsA ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ActiveBitsA?  
[:DigIO:ActiveBitsA ] 65535

**Description:** Query returns an integer corresponding to the "Active Bits" display on the "Carrier Status" tab on the Digital I/O panel. Each bit in the returned integer, if set, implies that the corresponding bit in "A" channel of the digital audio received signal has changed value at least once over a complete digital audio frame.

### ActiveBitsB?

**Command Syntax:** :DigIO:ActiveBitsB?

**Response Syntax:** [:DigIO:ActiveBitsB ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ActiveBitsB?  
[:DigIO:ActiveBitsB ] 65535

**Description:** Returns an integer corresponding to the "Active Bits" display on the "Carrier Status" tab on the Digital I/O panel. Each bit in the returned integer, if set, implies that the corresponding bit in the "B" channel of the digital audio received signal has changed value at least once over a complete digital audio block.

### DataBitsA?

**Command Syntax:** :DigIO:DataBitsA?

**Response Syntax:** [:DigIO:DataBitsA ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:DataBitsA?  
[:DigIO:DataBitsA ]128

**Description:** Returns an integer corresponding to the "Data Bits" display on the "Carrier Status" tab on the Digital I/O panel. Each bit in the returned integer, if set, implies that the corresponding bit in the "A" channel of the digital audio received signal was high in the first frame of the digital audio block.

## DataBitsB?

*Command Syntax:* :DigIO:DataBitsB?

*Response Syntax:* [:DigIO:DataBitsB ]*Value*

*Response Argument(s):* *Value* <int>

*Example:* :DigIO:DataBitsB?

[ :DigIO:DataBitsB ]128

*Description:* Returns an integer corresponding to the "Data Bits" display on the "Carrier Status" tab on the Digital I/O panel. Each bit in the returned integer, if set, implies that the corresponding bit in the "B" channel of the digital audio received signal was high in the first frame of the digital audio block.

## Channel Status Commands (General)

### SetTxChanStat

**Command Syntax:** :DigIO:SetTxChanStat Channel, Byte, Value

**Command Argument(s):** Channel <int> {chanA=0 | chanB=1}  
 Byte <int>  
 Value <int>

**Example:** :DigIO:SetTxChanStat chanA, 0, 4

**Description:** Sets the transmitted channel status for the given channel (chanA or chanB) and byte (0-23) to the value indicated.

### SetTxUserStat

**Command Syntax:** :DigIO:SetTxUserStat Channel, Byte, Value

**Command Argument (s):** Channel <int> {chanA=0 | chanB=1}  
 Byte <int>  
 Value <int>

**Example:** :DigIO:SetTxUserStat chanA, 0, 5

**Description:** Sets the transmitted user status for the given channel (chanA or chanB) and byte (0-23) to the value indicated.

### GetTxChanStat

**Command Syntax:** :DigIO:GetTxChanStat Channel, Byte

**Command Argument(s):** Channel <int> {chanA=0 | chanB=1}  
 Byte <int>

**Response Syntax:** [:DigIO:GetTxChanStat ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:GetTxChanStat chanA, 0  
 [:DigIO:GetTxChanStat ]7

**Description:** Queries the value of the transmitted channel status for the channel and byte indicated.

### GetTxUserStat

**Command Syntax:** :DigIO:GetTxUserStat Channel, Byte

**Command Argument(s):** Channel <int> {chanA=0 | chanB=1}  
 Byte <int>

**Response Syntax:** [:DigIO:GetTxUserStat ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:GetTxUserStat chanA, 0  
 [:DigIO:GetTxUserStat ]3

**Description:** Sets the value of the transmitted user status for the channel and byte indicated.

## TxChStatModeA

### TxChStatA

**Command Syntax:** :DigIO:TxChStatModeA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {csConsumer=0 | csProfessional=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:TxChStatModeA csConsumer

**Related Command(s):** TxChStatModeA?

**Description:** Sets the transmitted Consumer /Professional mode selection for channel A.

## TxChStatModeB

### TxChStatB

**Command Syntax:** :DigIO:TxChStatModeB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {csConsumer=0 | csProfessional=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:TxChStatModeB csProfessional

**Related Command(s):** TxChStatModeB?

**Description:** Sets the transmitted Consumer /Professional mode selection for channel A.

## TxChStatModeA?

### TxChStatA?

**Command Syntax:** :DigIO:TxChStatModeA?

**Response Syntax:** [:DigIO:TxChStatModeA ]*Value*

**Response Argument(s):** *Value* <int> {csConsumer=0 | csProfessional=1}

**Example:** :DigIO:TxChStatModeA?  
 [:DigIO:TxChStatModeA ]csConsumer

**Related Command(s):** TxChStatModeA

**Description:** Queries the transmitted Consumer /Professional mode selection for channel A.

## TxChStatModeB?

### TxChStatB?

**Command Syntax:** :DigIO:TxChStatModeB?

**Response Syntax:** [:DigIO:TxChStatModeB ]*Value*

**Response Argument(s):** *Value* <int> {csConsumer=0 | csProfessional=1}

**Example:** :DigIO:TxChStatModeB?  
 [:DigIO:TxChStatModeB ]csProfessional

**Related Command(s):** TxChStatModeB

**Description:** Queries the transmitted Consumer /Professional mode selection for channel B.



## TxChStatSelect? TxChStatSel?

**Command Syntax:** :DigIO:TxChStatSelect?

**Response Syntax:** [:DigIO:TxChStatSelect ]Value

**Response Argument(s):** Value <int> {chA=0 | chB=1 | chAB=2}

**Example:** :DigIO:TxChStatSelect?  
[:DigIO:TxChStatSelect ]chAB

**Related Command(s):** TxChStatSelect

**Description:** Queries the A, B, A/B selection at the top of the Channel Status Bits panel. (Note that this selection only effects the transmitted status selections made with the graphical user interface. When setting the channel status remotely it is not necessary to use this command.)

## TxChStatSelect TxChStatSel

**Command Syntax:** :DigIO:TxChStatSelect Value [, AllowCoercion]

**Command Argument(s):** Value <int> {chA=0 | chB=1 | chAB=2}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:TxChStatSelect Value [, AllowCoercion]

**Related Command(s):** TxChStatSelect?

**Description:** Sets the A, B, A/B selection at the top of the Channel Status Bits panel. (Note that this selection only effects the transmitted status selections made with the graphical user interface. When setting the channel status remotely it is not necessary to use this command.)

## GetRxChanStat

**Command Syntax:** :DigIO:GetRxChanStat Channel, Byte

**Command Argument(s):** Channel <int> {chanA=0 | chanB=1}  
Byte <int>

**Response Syntax:** [:DigIO:GetRxChanStat ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:GetRxChanStat chanA, 1  
[:DigIO:GetRxChanStat ]7

**Description:** Queries the indicate byte of the received channel status for the indicate channel.

## GetRxUserStat

**Command Syntax:** :DigIO:GetRxUserStat Channel, Byte

**Command Argument(s):** Channel <int> {chanA=0 | chanB=1}  
Byte <int>

**Response Syntax:** [:DigIO:GetRxUserStat ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:GetRxUserStat chanA, 0  
[:DigIO:GetRxUserStat ]0

## Transmitted Channel Status Commands (Professional)

### ProAAux?

**Command Syntax:** :DigIO:ProAAux?

**Response Syntax:** [:DigIO:ProAAux]Value

**Response Argument(s):** Value <int> {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}

**Example:** :DigIO:ProAAux?  
[:DigIO:ProAAux ]pa20bitNotDef

**Related Command(s):** ProAAux

**Description:** Queries the Auxilliary Bits setting of the channel A transmitted professional channel status.

### ProAAux

**Command Syntax:** :DigIO:ProAAux Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAAux pa24bitMainAud

**Related Command(s):** ProAAux?

**Description:** Sets the Auxilliary Bits selection of the channel A transmitted professional channel status.

### ProACh?

**Command Syntax:** :DigIO:ProACh?

**Response Syntax:** [:DigIO:ProACh]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:ProACh?  
[:DigIO:ProACh ]1

**Related Command(s):** ProACh

**Description:** Queries the channel # setting of the channel A transmitted professional channel status.

### ProACh

**Command Syntax:** :DigIO:ProACh Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProACh 2

**Related Command(s):** ProACh?

**Description:** Sets the channel # setting for the channel A transmitted professional channel status.

## ProAChMode?

**Command Syntax:** :DigIO:ProAChMode?

**Response Syntax:** [:DigIO:ProAChMode ]Value

**Response Argument(s):** Value <int> {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}

**Example:** :DigIO:ProAChMode?  
[:DigIO:ProAChMode ]pcmMono

**Related Command(s):** ProAChMode

**Description:** Queries the channel mode for the channel A transmitted professional channel status.

## ProAChMode

**Command Syntax:** :DigIO:ProAChMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAChMode pcmMono

**Related Command(s):** ProAChMode?

**Description:** Sets the channel mode for the channel A transmitted professional channel status.

## ProAConf0?

**Command Syntax:** :DigIO:ProAConf0?

**Response Syntax:** [:DigIO:ProAConf0 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAConf0?  
[:DigIO:ProAConf0 ]True

**Related Command(s):** ProAConf0

**Description:** Queries the status of the confidence (bytes 0-5) flag for the channel A transmitted professional channel status.

## ProAConf0

**Command Syntax:** :DigIO:ProAConf0 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAConf0 True

**Related Command(s):** ProAConf0?

**Description:** Sets the status of the confidence (bytes 0-5) flag for the channel A transmitted professional channel status.

## ProAConf1?

**Command Syntax:** :DigIO:ProAConf1?

**Response Syntax:** [:DigIO:ProAConf1 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAConf1?  
[:DigIO:ProAConf1 ]False

**Related Command(s):** ProAConf1

**Description:** Queries the status of the confidence (bytes 6-13) flag for the channel A transmitted professional channel status.

## ProAConf1

**Command Syntax:** :DigIO:ProAConf1 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAConf1 False

**Related Command(s):** ProAConf1?

**Description:** Sets the status of the confidence (bytes 6-13) flag for the channel A transmitted professional channel status.

## ProAConf2?

**Command Syntax:** :DigIO:ProAConf2?

**Response Syntax:** [:DigIO:ProAConf2 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAConf2?  
[:DigIO:ProAConf2 ]True

**Related Command(s):** ProAConf2

**Description:** Queries the status of the confidence (bytes 14-17) flag for the channel A transmitted professional channel status.

## ProAConf2

**Command Syntax:** :DigIO:ProAConf2 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAConf2 Value [, AllowCoercion]

**Related Command(s):** ProAConf2?

**Description:** Sets the status of the confidence (bytes 14-17) flag for the channel A transmitted professional channel status.

## ProAConf3?

**Command Syntax:** :DigIO:ProAConf3?

**Response Syntax:** [:DigIO:ProAConf3 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAConf3?

**[:DigIO:ProAConf3 ]False**

**Related Command(s):** ProAConf3

**Description:** Queries the status of the confidence (bytes 18-21) flag for the channel A transmitted professional channel status.

## ProAConf3

**Command Syntax:** :DigIO:ProAConf3 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProAConf3 Value [, AllowCoercion]

**Related Command(s):** ProAConf3?

**Description:** Sets the status of the confidence (bytes 18-21) flag for the channel A transmitted professional channel status.

## ProACrc?

**Command Syntax:** :DigIO:ProACrc?

**Response Syntax:** [:DigIO:ProACrc ]Value

**Response Argument(s):** Value <int> {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}

**Example:** :DigIO:ProACrc?

**[:DigIO:ProACrc ]pcCorrect**

**Related Command(s):** ProACrc

**Description:** Queries the CRC sending mode for the channel A transmitted professional channel status.

## ProACrc

**Command Syntax:** :DigIO:ProACrc Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProACrc pcIncorrect

**Related Command(s):** ProACrc?

**Description:** Sets the CRC sending mode for the channel A transmitted professional channel status.

## ProAData?

**Command Syntax:** :DigIO:ProAData?

**Response Syntax:** [:DigIO:ProAData ]Value

**Response Argument(s):** Value <int> {ddPCM=0|ddData=1}

**Example:** :DigIO:ProAData?

[:DigIO:ProAData] ddPCM

**Related Command(s):** ProAData

**Description:** Queries the PCM Audio/Data status for the channel A transmitted professional channel status.

## ProAData

**Command Syntax:** :DigIO:ProAData Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ddPCM=0|ddData=1}  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProAData ddPCM

**Related Command(s):** ProAData?

**Description:** Sets the PCM Audio/Data status for the channel A transmitted professional channel status.

## ProADest?

**Command Syntax:** :DigIO:ProADest?

**Response Syntax:** [:DigIO:ProADest ]Value

**Response Argument(s):** Value <string>

**Example:** :DigIO:ProADest?

[:DigIO:ProADest ]srs1

**Related Command(s):** ProADest

**Description:** Queries the 4 character destination label for the the channel A transmitted professional channel status.

## ProADest

**Command Syntax:** :DigIO:ProADest Value [, AllowCoercion]

**Command Argument(s):** Value <string>  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProADest srs1

**Related Command(s):** Setss the 4 character destination label for the the channel A transmitted professional channel status.

## ProAEmph?

**Command Syntax:** :DigIO:ProAEmph?

**Response Syntax:** [:DigIO:ProAEmph ]Value

**Response Argument(s):** Value <int> {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}

**Example:** :DigIO:ProAEmph?

[:DigIO:ProAEmph ]peNone

**Related Command(s):** ProAEmph

**Description:** Queries the emphasis mode selection for the channel A transmitted professional channel status.

## ProAEmph

**Command Syntax:** :DigIO:ProAEmph Value [, AllowCoercion]

**Command Argument(s):** Value <int> {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProAEmph peCCITT

**Related Command(s):** ProAEmph?

**Description:** Sets the emphasis mode selection for the channel A transmitted professional channel status.

## ProAFs?

**Command Syntax:** :DigIO:ProAFs?

**Response Syntax:** [:DigIO:ProAFs ]Value

**Response Argument(s):** Value <int> {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}

**Example:** :DigIO:ProAFs?

[:DigIO:ProAFs ]pf48k

**Related Command(s):** ProAFs

**Description:** Queries the sampling rate selection sent in the channel A transmitted professional channel status.

## ProAFs

**Command Syntax:** :DigIO:ProAFs Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProAFs pf192k

**Related Command(s):** ProAFs?

**Description:** Sets the sampling rate selection sent in the channel A transmitted professional channel status.

## ProAFsScl?

**Command Syntax:** :DigIO:ProAFsScl?

**Response Syntax:** [:DigIO:ProAFsScl]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAFsScl?  
[:DigIO:ProAFsScl]False

**Related Command(s):** ProAFsScl

**Description:** Queries the status of the "/1.001" flag sent with the channel A transmitted professional channel status.

## ProAFsScl

**Command Syntax:** :DigIO:ProAFsScl Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProAFsScl False

**Related Command(s):** ProAFsScl?

**Description:** Sets the status of the "/1.001" flag sent with the channel A transmitted professional channel status.

## ProAIncCode?

**Command Syntax:** :DigIO:ProAIncCode?

**Response Syntax:** [:DigIO:ProAIncCode]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProAIncCode?  
[:DigIO:ProAIncCode]False

**Related Command(s):** ProAIncCode

**Description:** Queries the status of the "Increment Local Address Code" selection for the channel A transmitted professional channel status.

## ProAIncCode

**Command Syntax:** :DigIO:ProAIncCode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProAIncCode True

**Related Command(s):** ProAIncCode?

**Description:** Sets the status of the "Increment Local Address Code" selection for the channel A transmitted professional channel status.



## ProAIncTime?

**Command Syntax:** :DigIO:ProAIncTime?

**Response Syntax:** [:DigIO:ProAIncTime ]Value

**Response Argument(s):** Value <int> {False=0|True=1}

**Example:** :DigIO:ProAIncTime?  
[:DigIO:ProAIncTime ]True

**Related Command(s):** ProAIncTime

**Description:** Queries the status of the "Increment Time Code" selection for the channel A transmitted professional channel status.

## ProAIncTime

**Command Syntax:** :DigIO:ProAIncTime Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0|True=1}  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProAIncTime True

**Related Command(s):** ProAIncTime?

**Description:** Sets the status of the "Increment Time Code" selection for the channel A transmitted professional channel status.

## ProALcl?

**Command Syntax:** :DigIO:ProALcl?

**Response Syntax:** [:DigIO:ProALcl ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:ProALcl?  
[:DigIO:ProALcl ]100

**Related Command(s):** ProALcl

**Description:** Queries the value of the "Local Address Code" sent with the channel A transmitted professional channel status.

## ProALcl

**Command Syntax:** :DigIO:ProALcl Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProALcl 100

**Related Command(s):** ProALcl?

**Description:** Sets the value of the "Local Address Code" sent with the channel A transmitted professional channel status.

## ProALocked?

**Command Syntax:** :DigIO:ProALocked?

**Response Syntax:** [:DigIO:ProALocked ]Value

**Response Argument(s):** Value <int> {plNotIndicated=0 | plpUnlocked=1}

**Example:** :DigIO:ProALocked?

[:DigIO:ProALocked ]plNotIndicated

**Related Command(s):** ProALocked

**Description:** Queries the status of the "Locked" selection for the channel A transmitted professional channel status.

## ProALocked

**Command Syntax:** :DigIO:ProALocked Value [, AllowCoercion]

**Command Argument(s):** Value <int> {plNotIndicated=0 | plpUnlocked=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProALocked plNotIndicated

**Related Command(s):** ProALocked?

**Description:** Sets the status of the "Locked" selection for the channel A transmitted professional channel status.

## ProAMCh?

**Command Syntax:** :DigIO:ProAMCh?

**Response Syntax:** [:DigIO:ProAMCh ]Value

**Response Argument(s):** Value <int> {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}

**Example:** :DigIO:ProAMCh?

[:DigIO:ProAMCh ]pcmMode0

**Related Command(s):** ProAMCh

**Description:** Queries the status of Multichannel Mode selection sent with channel A transmitted professional channel status.

## ProAMCh

**Command Syntax:** :DigIO:ProAMCh Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAMCh pmcUndef

**Related Command(s):** ProAMCh?

**Description:** Sets the status of Multichannel Mode selection sent with the channel A transmitted professional channel status.

## ProARef?

**Command Syntax:** :DigIO:ProARef?

**Response Syntax:** [:DigIO:ProARef ]Value

**Response Argument(s):** Value <int> {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}

**Example:** :DigIO:ProARef?  
[:DigIO:ProARef ]prGrade1

**Related Command(s):** ProARef

**Description:** Queries the Reference Signal selection of the channel A transmitted professional channel status.

## ProARef

**Command Syntax:** :DigIO:ProARef Value [, AllowCoercion]

**Command Argument(s):** Value <int> {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProARef prNotRef

**Related Command(s):** ProARef?

**Description:** Queries the Reference Signal selection of the channel A transmitted professional channel status.

## ProASrc?

**Command Syntax:** :DigIO:ProASrc?

**Response Syntax:** [:DigIO:ProASrc ]Value

**Response Argument(s):** Value <string>

**Example:** :DigIO:ProASrc?  
[:DigIO:ProASrc ]srs2

**Related Command(s):** ProASrc

**Description:** Queries the 4 character Source Label sent with the channel A transmitted professional channel status.

## ProASrc

**Command Syntax:** :DigIO:ProASrc Value [, AllowCoercion]

**Command Argument(s):** Value <string>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProASrc srs2, True

**Related Command(s):** ProASrc?

**Description:** Sets the 4 character Source Label sent with the channel A transmitted professional channel status.

## ProATime?

**Command Syntax:** :DigIO:ProATime?

**Response Syntax:** [:DigIO:ProATime ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:ProATime?  
[:DigIO:ProATime ]100

**Related Command(s):** ProATime

**Description:** Queries the Time Code sent with the channel A transmitted professional channel status.

## ProATime

**Command Syntax:** :DigIO:ProATime Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProATime 100

**Related Command(s):** ProATime?

**Description:** Sets the Time Code sent with the channel A transmitted professional channel status.

## ProAUser?

**Command Syntax:** :DigIO:ProAUser?

**Response Syntax:** [:DigIO:ProAUser ]Value

**Response Argument(s):** Value <int> {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4  
| puRsvdForMetadata=5}

**Example:** :DigIO:ProAUser?  
[:DigIO:ProAUser ]puNoUserInfo

**Related Command(s):** ProAUser

**Description:** Queries the User Bits mode sent with the channel A transmitted professional channel status.

## ProAUser

**Command Syntax:** :DigIO:ProAUser Value [, AllowCoercion]

**Command Argument(s):** Value <int> {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4  
| puRsvdForMetadata=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAUser puAES18

**Related Command(s):** ProAUser?

**Description:** Sets the User Bits mode sent with the channel A transmitted professional channel status.

## ProAWdLen?

**Command Syntax:** :DigIO:ProAWdLen?

**Response Syntax:** [:DigIO:ProAWdLen ]Value

**Response Argument(s):** Value <int> {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}

**Example:** :DigIO:ProAWdLen?  
[:DigIO:ProAWdLen ]pwl24

**Related Command(s):** ProAWdLen

**Description:** Queries the Word Length indication sent with the channel A transmitted professional channel status.

## ProAWdLen

**Command Syntax:** :DigIO:ProAWdLen Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProAWdLen Value [, AllowCoercion]

**Related Command(s):** ProAWdLen?

**Description:** Sets the Word Length indication sent with the channel A transmitted professional channel status.

## ProBAux?

**Command Syntax:** :DigIO:ProBAux?

**Response Syntax:** [:DigIO:ProBAux ]Value

**Response Argument(s):** Value <int> {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}

**Example:** :DigIO:ProBAux?  
[:DigIO:ProBAux ]pa20bitNotDef

**Related Command(s):** ProBAux

**Description:** Queries the Auxilliary Bits setting of the channel B transmitted professional channel status.

## ProBAux

**Command Syntax:** :DigIO:ProBAux Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBAux pa24bitMainAud

**Related Command(s):** ProBAux?

**Description:** Sets the Auxilliary Bits selection of the channel B transmitted professional channel status.

## ProBCh?

**Command Syntax:** :DigIO:ProBCh?

**Response Syntax:** [:DigIO:ProBCh ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ProBCh?  
[:DigIO:ProBCh ]1

**Related Command(s):** ProBCh

**Description:** Queries the channel # setting of the channel B transmitted professional channel status.

## ProBCh

**Command Syntax:** :DigIO:ProBCh *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ProBCh 2

**Related Command(s):** ProBCh?

**Description:** Sets the channel # setting for the channel B transmitted professional channel status.

## ProBChMode?

**Command Syntax:** :DigIO:ProBChMode?

**Response Syntax:** [:DigIO:ProBChMode ]*Value*

**Response Argument(s):** *Value* <int> {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}

**Example:** :DigIO:ProBChMode?  
[:DigIO:ProBChMode ]pcmMono

**Related Command(s):** ProBChMode

**Description:** Queries the channel mode for the channel B transmitted professional channel status.

## ProBChMode

**Command Syntax:** :DigIO:ProBChMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ProBChMode pcmMono

**Related Command(s):** ProBChMode?

**Description:** Sets the channel mode for the channel B transmitted professional channel status.

## ProBConf0?

**Command Syntax:** :DigIO:ProBConf0?

**Response Syntax:** [:DigIO:ProBConf0 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProBConf0?  
[:DigIO:ProBConf0 ]True

**Related Command(s):** ProBConf0

**Description:** Queries the status of the confidence (bytes 0-5) flag for the channel B transmitted professional channel status.

## ProBConf0

**Command Syntax:** :DigIO:ProBConf0 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBConf0 True

**Related Command(s):** ProBConf0?

**Description:** Sets the status of the confidence (bytes 0-5) flag for the channel B transmitted professional channel status.

## ProBConf1?

**Command Syntax:** :DigIO:ProBConf1?

**Response Syntax:** [:DigIO:ProBConf1 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProBConf1?  
[:DigIO:ProBConf1 ]False

**Related Command(s):** ProBConf1

**Description:** Queries the status of the confidence (bytes 6-13) flag for the channel B transmitted professional channel status.

## ProBConf1

**Command Syntax:** :DigIO:ProBConf1 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBConf1 False

**Related Command(s):** ProBConf1?

**Description:** Sets the status of the confidence (bytes 6-13) flag for the channel B transmitted professional channel status.

## ProBConf2?

**Command Syntax:** :DigIO:ProBConf2?

**Response Syntax:** [:DigIO:ProBConf2 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProBConf2?  
[:DigIO:ProBConf2 ]True

**Related Command(s):** ProBConf2

**Description:** Queries the status of the confidence (bytes 14-17) flag for the channel B transmitted professional channel status.

## ProBConf2

**Command Syntax:** :DigIO:ProBConf2 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBConf2 Value [, AllowCoercion]

**Related Command(s):** ProBConf2?

**Description:** Sets the status of the confidence (bytes 14-17) flag for the channel B transmitted professional channel status.

## ProBConf3?

**Command Syntax:** :DigIO:ProBConf3?

**Response Syntax:** [:DigIO:ProBConf3 ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProBConf3?  
[:DigIO:ProBConf3 ]False

**Related Command(s):** ProBConf3

**Description:** Queries the status of the confidence (bytes 18-21) flag for the channel B transmitted professional channel status.

## ProBConf3

**Command Syntax:** :DigIO:ProBConf3 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBConf3 Value [, AllowCoercion]

**Related Command(s):** ProBConf3?

**Description:** Sets the status of the confidence (bytes 18-21) flag for the channel B transmitted professional channel status.



## ProBCrc?

**Command Syntax:** :DigIO:ProBCrc?

**Response Syntax:** [:DigIO:ProBCrc ]*Value*

**Response Argument(s):** *Value* <int> {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}

**Example:** :DigIO:ProBCrc?  
[:DigIO:ProBCrc ]pcCorrect

**Related Command(s):** ProBCrc

**Description:** Queries the CRC sending mode for the channel B transmitted professional channel status.

## ProBCrc

**Command Syntax:** :DigIO:ProBCrc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Default:** *Value*

**Example:** :DigIO:ProBCrc pcIncorrect

**Related Command(s):** ProBCrc?

**Description:** Sets the CRC sending mode for the channel B transmitted professional channel status.

## ProBData?

**Command Syntax:** :DigIO:ProBData?

**Response Syntax:** [:DigIO:ProBData ]*Value*

**Response Argument(s):** *Value* <int> {ddPCM=0 | ddData=1}

**Example:** :DigIO:ProBData?  
[:DigIO:ProBData] ddPCM

**Related Command(s):** ProBData

**Description:** Queries the PCM Audio/Data status for the channel B transmitted professional channel status.

## ProBData

**Command Syntax:** :DigIO:ProBData *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ddPCM=0 | ddData=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Default:** *Value*

**Example:** :DigIO:ProBData ddPCM

**Related Command(s):** ProBData?

**Description:** Sets the PCM Audio/Data status for the channel B transmitted professional channel status.

## ProBDest?

**Command Syntax:** :DigIO:ProBDest?

**Response Syntax:** [:DigIO:ProBDest ]Value

**Response Argument(s):** Value <string>

**Example:** :DigIO:ProBDest?  
[:DigIO:ProBDest ]srs1

**Related Command(s):** ProBDest

**Description:** Queries the 4 character destination label for the the channel B transmitted professional channel status.

## ProBDest

**Command Syntax:** :DigIO:ProBDest Value [, AllowCoercion]

**Command Argument(s):** Value <string>

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBDest srs1

**Related Command(s):** Setss the 4 character destination label for the the channel B transmitted professional channel status.

## ProBEmph?

**Command Syntax:** :DigIO:ProBEmph?

**Response Syntax:** [:DigIO:ProBEmph ]Value

**Response Argument(s):** Value <int> {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}

**Example:** :DigIO:ProBEmph?  
[:DigIO:ProBEmph ]peNone

**Related Command(s):** ProBEmph

**Description:** Queries the emphasis mode selection for the channel B transmitted professional channel status.

## ProBEmph

**Command Syntax:** :DigIO:ProBEmph Value [, AllowCoercion]

**Command Argument(s):** Value <int> {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBEmph peCCITT

**Related Command(s):** ProBEmph?

**Description:** Sets the emphasis mode selection for the channel B transmitted professional channel status.

## ProBFs?

**Command Syntax:** :DigIO:ProBFs?

**Response Syntax:** [:DigIO:ProBFs ]Value

**Response Argument(s):** Value <int> {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}

**Example:** :DigIO:ProBFs?  
[:DigIO:ProBFs ]pf48k

**Related Command(s):** ProBFs

**Description:** Queries the sampling rate selection sent in the channel B transmitted professional channel status.

## ProBFs

**Command Syntax:** :DigIO:ProBFs Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBFs pf192k

**Related Command(s):** ProBFs?

**Description:** Sets the sampling rate selection sent in the channel B transmitted professional channel status.

## ProBFsScI?

**Command Syntax:** :DigIO:ProBFsScI?

**Response Syntax:** [:DigIO:ProBFsScI ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ProBFsScI?  
[:DigIO:ProBFsScI ]False

**Related Command(s):** ProBFsScI

**Description:** Queries the status of the "/1.001" flag sent with the channel B transmitted professional channel status.

## ProBFsScI

**Command Syntax:** :DigIO:ProBFsScI Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBFsScI False

**Related Command(s):** ProBFsScI?

**Description:** Sets the status of the "/1.001" flag sent with the channel B transmitted professional channel status.

## ProBIncCode?

**Command Syntax:** :DigIO:ProBIncCode?

**Response Syntax:** [:DigIO:ProBIncCode ]Value

**Response Argument(s):** Value <int> {False=0|True=1}

**Example:** :DigIO:ProBIncCode?  
[:DigIO:ProBIncCode ]False

**Related Command(s):** ProBIncCode

**Description:** Queries the status of the "Increment Local Address Code" selection for the channel B transmitted professional channel status.

## ProBIncCode

**Command Syntax:** :DigIO:ProBIncCode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0|True=1}  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProBIncCode True

**Related Command(s):** ProBIncCode?

**Description:** Sets the status of the "Increment Local Address Code" selection for the channel B transmitted professional channel status.

## ProBIncTime?

**Command Syntax:** :DigIO:ProBIncTime?

**Response Syntax:** [:DigIO:ProBIncTime ]Value

**Response Argument(s):** Value <int> {False=0|True=1}

**Example:** :DigIO:ProBIncTime?  
[:DigIO:ProBIncTime ]True

**Related Command(s):** ProBIncTime

**Description:** Queries the status of the "Increment Time Code" selection for the channel B transmitted professional channel status.

## ProBIncTime

**Command Syntax:** :DigIO:ProBIncTime Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0|True=1}  
AllowCoercion <bool> {False=0|True=1}

**Default:** Value

**Example:** :DigIO:ProBIncTime True

**Related Command(s):** ProBIncTime?

**Description:** Sets the status of the "Increment Time Code" selection for the channel B transmitted professional channel status.

## ProBLcl?

**Command Syntax:** :DigIO:ProBLcl?

**Response Syntax:** [:DigIO:ProBLcl ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:ProBLcl?  
[:DigIO:ProBLcl ]100

**Related Command(s):** ProBLcl

**Description:** Queries the value of the "Local Address Code" sent with the channel B transmitted professional channel status.

## ProBLcl

**Command Syntax:** :DigIO:ProBLcl Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBLcl 100

**Related Command(s):** ProBLcl?

**Description:** Sets the value of the "Local Address Code" sent with the channel B transmitted professional channel status.

## ProBLocked?

**Command Syntax:** :DigIO:ProBLocked?

**Response Syntax:** [:DigIO:ProBLocked ]Value

**Response Argument(s):** Value <int> {plNotIndicated=0 | plpUnlocked=1}

**Example:** :DigIO:ProBLocked?  
[:DigIO:ProBLocked ]plNotIndicated

**Related Command(s):** ProBLocked

**Description:** Queries the status of the "Locked" selection for the channel B transmitted professional channel status.

## ProBLocked

**Command Syntax:** :DigIO:ProBLocked Value [, AllowCoercion]

**Command Argument(s):** Value <int> {plNotIndicated=0 | plpUnlocked=1}

AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ProBLocked plNotIndicated

**Related Command(s):** ProBLocked?

**Description:** Sets the status of the "Locked" selection for the channel B transmitted professional channel status.

## ProBMCh?

**Command Syntax:** :DigIO:ProBMCh?

**Response Syntax:** [:DigIO:ProBMCh ]Value

**Response Argument(s):** Value <int> {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}

**Example:** :DigIO:ProBMCh?  
[:DigIO:ProBMCh ]pcmMode0

**Related Command(s):** ProBMCh

**Description:** Queries the status of Multichannel Mode selection sent with channel B transmitted professional channel status.

## ProBMCh

**Command Syntax:** :DigIO:ProBMCh Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBMCh pmcUndef

**Related Command(s):** ProBMCh?

**Description:** Sets the status of Multichannel Mode selection sent with the channel B transmitted professional channel status.

## ProBRef?

**Command Syntax:** :DigIO:ProBRef?

**Response Syntax:** [:DigIO:ProBRef ]Value

**Response Argument(s):** Value <int> {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}

**Example:** :DigIO:ProBRef?  
[:DigIO:ProBRef ]prGrade1

**Related Command(s):** ProBRef

**Description:** Queries the Reference Signal selection of the channel B transmitted professional channel status.

## ProBRef

**Command Syntax:** :DigIO:ProBRef Value [, AllowCoercion]

**Command Argument(s):** Value <int> {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBRef prNotRef

**Related Command(s):** ProBRef?

**Description:** Queries the Reference Signal selection of the channel B transmitted professional channel status.

## ProBSrc?

**Command Syntax:** :DigIO:ProBSrc?

**Response Syntax:** [:DigIO:ProBSrc ]*Value*

**Response Argument(s):** *Value* <string>

**Example:** :DigIO:ProBSrc?  
[:DigIO:ProBSrc ]srs2

**Related Command(s):** ProBSrc

**Description:** Queries the 4 character Source Label sent with the channel B transmitted professional channel status.

## ProBSrc

**Command Syntax:** :DigIO:ProBSrc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <string>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ProBSrc srs2, True

**Related Command(s):** ProBSrc?

**Description:** Sets the 4 character Source Label sent with the channel B transmitted professional channel status.

## ProBTime?

**Command Syntax:** :DigIO:ProBTime?

**Response Syntax:** [:DigIO:ProBTime ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ProBTime?  
[:DigIO:ProBTime ]100

**Related Command(s):** ProBTime

**Description:** Queries the Time Code sent with the channel B transmitted professional channel status.

## ProBTime

**Command Syntax:** :DigIO:ProBTime *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ProBTime 100

**Related Command(s):** ProBTime?

**Description:** Sets the Time Code sent with the channel B transmitted professional channel status.

## ProBUser?

**Command Syntax:** :DigIO:ProBUser?

**Response Syntax:** [:DigIO:ProBUser ]Value

**Response Argument(s):** Value <int> {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4 | puRsvdForMetadata=5}

**Example:** :DigIO:ProBUser?  
[:DigIO:ProBUser ]puNoUserInfo

**Related Command(s):** ProBUser

**Description:** Queries the User Bits mode sent with the channel B transmitted professional channel status.

## ProBUser

**Command Syntax:** :DigIO:ProBUser Value [, AllowCoercion]

**Command Argument(s):** Value <int> {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4 | puRsvdForMetadata=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBUser puAES18

**Related Command(s):** ProBUser?

**Description:** Sets the User Bits mode sent with the channel B transmitted professional channel status.

## ProBWdLen?

**Command Syntax:** :DigIO:ProBWdLen?

**Response Syntax:** [:DigIO:ProBWdLen ]Value

**Response Argument(s):** Value <int> {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}

**Example:** :DigIO:ProBWdLen?  
[:DigIO:ProBWdLen ]pwl24

**Related Command(s):** ProBWdLen

**Description:** Queries the Word Length indication sent with the channel B transmitted professional channel status.

## ProBWdLen

**Command Syntax:** :DigIO:ProBWdLen Value [, AllowCoercion]

**Command Argument(s):** Value <int> {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ProBWdLen Value [, AllowCoercion]

**Related Command(s):** ProBWdLen?

**Description:** Sets the Word Length indication sent with the channel B transmitted professional channel status.



## Channel Status Commands(Consumer)

### ConACat?

**Command Syntax:** :DigIO:ConACat?

**Response Syntax:** [:DigIO:ConACat ]*Value*

**Response Argument(s):** *Value* <int> {catGeneral=0 | catLaser=1 | catCD=2 | catMinidisc=3 | catDVD=4 | catPCM=5 | catMixer=6 | catFsConvert=7 | catSndSampler=8 | catSndProc=9 | catDAT=10 | catVideoTape=11 | catDCC=12 | catElectronicSoftware=13 | catSynth=14 | catMic=15 | catADCnoCopyrt=16 | catADCwCopyrt=17 | catSolidState=18 | catExperimental=19}

**Example:** :DigIO:ConACat?

**[ :DigIO:ConACat ]catLaser**

**Related Command(s):** ConACat

**Description:** Queries the consumer category code indicated by the Channel A digital audio received status bits.

### ConACat

**Command Syntax:** :DigIO:ConACat *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {catGeneral=0 | catLaser=1 | catCD=2 | catMinidisc=3 | catDVD=4 | catPCM=5 | catMixer=6 | catFsConvert=7 | catSndSampler=8 | catSndProc=9 | catDAT=10 | catVideoTape=11 | catDCC=12 | catElectronicSoftware=13 | catSynth=14 | catMic=15 | catADCnoCopyrt=16 | catADCwCopyrt=17 | catSolidState=18 | catExperimental=19}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConACat catCD

**Related Command(s):** ConACat?

**Description:** Sets the consumer category code transmitted in the Channel A digital audio status bits.

### ConACh?

**Command Syntax:** :DigIO:ConACh?

**Response Syntax:** [:DigIO:ConACh ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ConACh?

**[ :DigIO:ConACh ]1**

**Related Command(s):** ConACh

**Description:** Queries the Channel A consumer Channel #. (0 = N/A, 1=A, 2=B, etc.)

### ConACh

**Command Syntax:** :DigIO:ConACh *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConACh 1

**Related Command(s):** ConACh?

**Description:** Sets the Channel A transmitted consumer Channel # field.

## ConAClkAcc?

**Command Syntax:** :DigIO:ConAClkAcc?

**Response Syntax:** [:DigIO:ConAClkAcc ]Value

**Response Argument(s):** Value <int> {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateFs=3}

**Example:** :DigIO:ConAClkAcc?

**[:DigIO:ConAClkAcc ] ccaLevel2**

**Related Command(s):** ConAClkAcc

**Description:** Queries the Channel A received consumer Clock Accuracy field

## ConAClkAcc

**Command Syntax:** :DigIO:ConAClkAcc Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateFs=3}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConAClkAcc 2

**Related Command(s):** ConAClkAcc?

**Description:** Sets the Channel A transmitted consumer Clock Accuracy field.

## ConACopyright?

**Command Syntax:** :DigIO:ConACopyright?

**Response Syntax:** [:DigIO:ConACopyright ]Value

**Response Argument(s):** Value <int> {ccCopyright=0 | ccNoCopyright=1}

**Example:** :DigIO:ConACopyright?

**[:DigIO:ConACopyright ] ccNoCopyright**

**Related Command(s):** ConACopyright

**Description:** Queries the Channel A consumer received Copyright Bit.

## ConACopyright

**Command Syntax:** :DigIO:ConACopyright Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ccCopyright=0 | ccNoCopyright=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConACopyright false

**Related Command(s):** ConACopyright?

**Description:** Sets the Channel A consumer transmit Copyright Bit.

## ConAData?

**Command Syntax:** :DigIO:ConAData?

**Response Syntax:** [:DigIO:ConAData ]Value

**Response Argument(s):** Value <int> {ddPCM=0 | ddData=1}

**Example:** :DigIO:ConAData?  
[:DigIO:ConAData ]ddPCM

**Related Command(s):** ConAData

**Description:** Queries the Channel A consumer received Audio Sample field.

## ConAData

**Command Syntax:** :DigIO:ConAData Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ddPCM=0 | ddData=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConAData Value [, AllowCoercion]

**Related Command(s):** ConAData?

**Description:** Sets the Channel A consumer transmitted Audio Sample field.

## ConAEmph?

**Command Syntax:** :DigIO:ConAEmph?

**Response Syntax:** [:DigIO:ConAEmph ]Value

**Response Argument(s):** Value <int> {ceNone=0 | ce5015=1}

**Example:** :DigIO:ConAEmph?  
[:DigIO:ConAEmph ]ceNone

**Related Command(s):** ConAEmph

**Description:** Queries the Channel A consumer received Emphasis field.

## ConAEmph

**Command Syntax:** :DigIO:ConAEmph Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ceNone=0 | ce5015=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConAEmph ceNone

**Related Command(s):** ConAEmph?

**Description:** Sets the Channel A consumer transmitted EmphasisField

## ConAFs?

**Command Syntax:** :DigIO:ConAFs?

**Response Syntax:** [:DigIO:ConAFs ]Value

**Response Argument(s):** Value <int> {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}

**Example:** :DigIO:ConAFs?  
[:DigIO:ConAFs ] cf48k

**Related Command(s):** ConAFs

**Description:** Queries the Channel A consumer received Sampling Frequency field.

## ConAFs

**Command Syntax:** `:DigIO:ConAFs Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigIO:ConAFs Value [, AllowCoercion]`

**Related Command(s):** ConAFs?

**Description:** Queries the Channel A consumer received Sampling Frequency field.

## ConAFsOrig?

**Command Syntax:** `:DigIO:ConAFsOrig?`

**Response Syntax:** `[:DigIO:ConAFsOrig ]Value`

**Response Argument(s):** `Value <int> {cof44k=0 | cof88k=1 | cof22k=2 | cof176k=3 | cof48k=4 | cof96k=5 | cof24k=6 | cof192k=7 | cofRsvd0=8 | cof8k=9 | cof11k=10 | cof12k=11 | cof32k=12 | cofRsvd1=13 | cof16k=14 | cofNotIndicated=15}`

**Example:** `:DigIO:ConAFsOrig?`

`[:DigIO:ConAFsOrig ]cof44k`

**Related Command(s):** ConAFsOrig

**Description:** Queries the Channel A consumer received "Original Fs" field.

## ConAFsOrig

**Command Syntax:** `:DigIO:ConAFsOrig Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {cof44k=0 | cof88k=1 | cof22k=2 | cof176k=3 | cof48k=4 | cof96k=5 | cof24k=6 | cof192k=7 | cofRsvd0=8 | cof8k=9 | cof11k=10 | cof12k=11 | cof32k=12 | cofRsvd1=13 | cof16k=14 | cofNotIndicated=15}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigIO:ConAFsOrig cof44k`

**Related Command(s):** ConAFsOrig?

**Description:** Sets the Channel A consumer transmitted "Original Fs" field.

## ConAPreRec?

**Command Syntax:** `:DigIO:ConAPreRec?`

**Response Syntax:** `[:DigIO:ConAPreRec ]Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigIO:ConAPreRec?`

`[:DigIO:ConAPreRec ]True`

**Related Command(s):** ConAPreRec

**Description:** Queries the Channel A consumer received "Pre-Recorded" status.

## ConAPreRec

**Command Syntax:** :DigIO:ConAPreRec *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConAPreRec True

**Related Command(s):** ConAPreRec?

**Description:** Sets the Channel A consumer transmitted "Pre-Recorded" status.

## ConASrc?

**Command Syntax:** :DigIO:ConASrc?

**Response Syntax:** [:DigIO:ConASrc ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ConASrc?  
 [:DigIO:ConASrc ]3

**Related Command(s):** ConASrc

**Description:** Queries the Channel A consumer received "Source #" field.

## ConASrc

**Command Syntax:** :DigIO:ConASrc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConASrc 3

**Related Command(s):** ConASrc?

**Description:** Sets the Channel A consumer transmitted "Source #" field.

## ConAWdLen?

**Command Syntax:** :DigIO:ConAWdLen?

**Response Syntax:** [:DigIO:ConAWdLen ]*Value*

**Response Argument(s):** *Value* <int> {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 |  
 cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}

**Example:** :DigIO:ConAWdLen?  
 [:DigIO:ConAWdLen ]cwa24

**Related Command(s):** ConAWdLen

**Description:** Queries the Channel A consumer received Word Length field.

## ConAWdLen

**Command Syntax:** :DigIO:ConAWdLen *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 |  
 cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConAWdLen *Value* [, *AllowCoercion*]

**Related Command(s):** ConAWdLen?

**Description:** Sets the Channel A consumer transmitted Word Length field.

## ConBCat

**Command Syntax:** :DigIO:ConBCat *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {catGeneral=0 | catLaser=1 | catCD=2 | catMinidisc=3 | catDVD=4 | catPCM=5 | catMixer=6 | catFsConvert=7 | catSndSampler=8 | catSndProc=9 | catDAT=10 | catVideoTape=11 | catDCC=12 | catElectronicSoftware=13 | catSynth=14 | catMic=15 | catADCnoCopyrt=16 | catADCwCopyrt=17 | catSolidState=18 | catExperimental=19}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConACat catCD

**Related Command(s):** ConBCat?

**Description:** Sets the consumer category code transmitted in the Channel B digital audio status bits.

## ConBCh?

**Command Syntax:** :DigIO:ConBCh?

**Response Syntax:** [:DigIO:ConBCh ]*Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigIO:ConBCh?  
 [:DigIO:ConBCh ]1

**Related Command(s):** ConBCh

**Description:** Queries the Channel B consumer Channel #. (0 = N/A, 1=A, 2=B, etc.)

## ConBCh

**Command Syntax:** :DigIO:ConBCh *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBCh 1

**Related Command(s):** ConBCh?

**Description:** Sets the Channel B transmitted consumer Channel # field.

## ConBclkAcc?

**Command Syntax:** :DigIO:ConBclkAcc?

**Response Syntax:** [:DigIO:ConBclkAcc ]*Value*

**Response Argument(s):** *Value* <int> {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateFs=3}

**Example:** :DigIO:ConBclkAcc?  
 [:DigIO:ConBclkAcc ] ccaLevel2

**Related Command(s):** ConBclkAcc

**Description:** Queries the Channel B received consumer Clock Accuracy field

## ConBCLKAcc

**Command Syntax:** :DigIO:ConBCLKAcc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateNfs=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBCLKAcc 2

**Related Command(s):** ConBCLKAcc?

**Description:** Sets the Channel B transmitted consumer Clock Accuracy field.

## ConBCopyright?

**Command Syntax:** :DigIO:ConBCopyright?

**Response Syntax:** [:DigIO:ConBCopyright ]*Value*

**Response Argument(s):** *Value* <int> {ccCopyright=0 | ccNoCopyright=1}

**Example:** :DigIO:ConBCopyright?

**[:DigIO:ConBCopyright ] ccNoCopyright**

**Related Command(s):** ConBCopyright

**Description:** Queries the Channel B consumer received Copyright Bit.

## ConBCopyright

**Command Syntax:** :DigIO:ConBCopyright *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ccCopyright=0 | ccNoCopyright=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBCopyright false

**Related Command(s):** ConBCopyright?

**Description:** Sets the Channel B consumer transmit Copyright Bit.

## ConBData?

**Command Syntax:** :DigIO:ConBData?

**Response Syntax:** [:DigIO:ConBData ]*Value*

**Response Argument(s):** *Value* <int> {ddPCM=0 | ddData=1}

**Example:** :DigIO:ConBData?

**[:DigIO:ConBData ]ddPCM**

**Related Command(s):** ConBData

**Description:** Queries the Channel B consumer received Audio Sample field.

## ConBData

**Command Syntax:** :DigIO:ConBData *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ddPCM=0 | ddData=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBData *Value* [, *AllowCoercion*]

**Related Command(s):** ConBData?

**Description:** Sets the Channel B consumer transmitted Audio Sample field.

## ConBEmph?

**Command Syntax:** :DigIO:ConBEmph?

**Response Syntax:** [:DigIO:ConBEmph ]Value

**Response Argument(s):** Value <int> {ceNone=0 | ce5015=1}

**Example:** :DigIO:ConBEmph?

**[ :DigIO:ConBEmph ] ceNone**

**Related Command(s):** ConBEmph

**Description:** Queries the Channel B consumer received Emphasis field.

## ConBEmph

**Command Syntax:** :DigIO:ConBEmph Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ceNone=0 | ce5015=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConBEmph ceNone

**Related Command(s):** ConBEmph?

**Description:** Sets the Channel B consumer transmitted EmphasisField

## ConBFs?

**Command Syntax:** :DigIO:ConBFs?

**Response Syntax:** [:DigIO:ConBFs ]Value

**Response Argument(s):** Value <int> {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}

**Example:** :DigIO:ConBFs?

**[ :DigIO:ConBFs ] cf48k**

**Related Command(s):** ConBFs

**Description:** Queries the Channel B consumer received Sampling Frequency field.

## ConBFs

**Command Syntax:** :DigIO:ConBFs Value [, AllowCoercion]

**Command Argument(s):** Value <int> {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConBFs Value [, AllowCoercion]

**Related Command(s):** ConBFs?

**Description:** Queries the Channel B consumer received Sampling Frequency field.



## ConBFsOrig?

**Command Syntax:** :DigIO:ConBFsOrig?

**Response Syntax:** [:DigIO:ConBFsOrig ]*Value*

**Response Argument(s):** *Value* <int> {cof44k=0 | cof88k=1 | cof22k=2 | cof176k=3 | cof48k=4 | cof96k=5 | cof24k=6 | cof192k=7 | cofRsvd0=8 | cof8k=9 | cof11k=10 | cof12k=11 | cof32k=12 | cofRsvd1=13 | cof16k=14 | cofNotIndicated=15}

**Example:** :DigIO:ConBFsOrig?  
[:DigIO:ConBFsOrig ]cof44k

**Related Command(s):** ConBFsOrig

**Description:** Queries the Channel B consumer received "Original Fs" field.

## ConBFsOrig

**Command Syntax:** :DigIO:ConBFsOrig *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cof44k=0 | cof88k=1 | cof22k=2 | cof176k=3 | cof48k=4 | cof96k=5 | cof24k=6 | cof192k=7 | cofRsvd0=8 | cof8k=9 | cof11k=10 | cof12k=11 | cof32k=12 | cofRsvd1=13 | cof16k=14 | cofNotIndicated=15}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBFsOrig cof44k

**Related Command(s):** ConBFsOrig?

**Description:** Sets the Channel B consumer transmitted "Original Fs" field.

## ConBPreRec?

**Command Syntax:** :DigIO:ConBPreRec?

**Response Syntax:** [:DigIO:ConBPreRec ]*Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigIO:ConBPreRec?  
[:DigIO:ConBPreRec ]True

**Related Command(s):** ConBPreRec

**Description:** Queries the Channel B consumer received "Pre-Recorded" status.

## ConBPreRec

**Command Syntax:** :DigIO:ConBPreRec *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigIO:ConBPreRec True

**Related Command(s):** ConBPreRec?

**Description:** Sets the Channel B consumer transmitted "Pre-Recorded" status.

## ConBSrc?

**Command Syntax:** :DigIO:ConBSrc?

**Response Syntax:** [:DigIO:ConBSrc ]Value

**Response Argument(s):** Value <int>

**Example:** :DigIO:ConBSrc?  
[:DigIO:ConBSrc ]3

**Related Command(s):** ConBSrc

**Description:** Queries the Channel B consumer received "Source #" field.

## ConBSrc

**Command Syntax:** :DigIO:ConBSrc Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConBSrc 3

**Related Command(s):** ConBSrc?

**Description:** Sets the Channel B consumer transmitted "Source #" field.

## ConBWdLen?

**Command Syntax:** :DigIO:ConBWdLen?

**Response Syntax:** [:DigIO:ConBWdLen ]Value

**Response Argument(s):** Value <int> {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 | cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}

**Example:** :DigIO:ConBWdLen?  
[:DigIO:ConBWdLen ]cwa24

**Related Command(s):** ConBWdLen

**Description:** Queries the Channel B consumer received Word Length field.

## ConBWdLen

**Command Syntax:** :DigIO:ConBWdLen Value [, AllowCoercion]

**Command Argument(s):** Value <int> {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 | cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ConBWdLen Value [, AllowCoercion]

**Related Command(s):** ConBWdLen?

**Description:** Sets the Channel B consumer transmitted Word Length field.

## Impairment Commands

### ClockOutJitter?

**Command Syntax:** :DigIO:ClockOutJitter?

**Response Syntax:** [:DigIO:ClockOutJitter]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ClockOutJitter?  
[:DigIO:ClockOutJitter]False

**Related Command(s):** ClockOutJitter

**Description:** Query that returns the value of the Rear Panel Clock Jitter Enable.

### ClockOutJitter

**Command Syntax:** :DigIO:ClockOutJitter Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ClockOutJitter True

**Related Command(s):** ClockOutJitter?

**Description:** Sets the value of the Rear Panel Clock Jitter Enable

### ImpairCableSim?

**Command Syntax:** :DigIO:ImpairCableSim?

**Response Syntax:** [:DigIO:ImpairCableSim]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ImpairCableSim?  
[:DigIO:ImpairCableSim]False

**Related Command(s):** ImpairCableSim

**Description:** Queries the status of the Cable Simulator on the Output Impairment tab of the Digital I/O panel.

### ImpairCableSim

**Command Syntax:** :DigIO:ImpairCableSim Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Default:** Value

**Example:** :DigIO:ImpairCableSim True

**Related Command(s):** ImpairCableSim?

**Description:** Sets the Cable Simulator status on the Output Impairment tab of the Digital I/O panel.

## ImpairCarrierInvert?

**Command Syntax:** :DigIO:ImpairCarrierInvert?

**Response Syntax:** [:DigIO:ImpairCarrierInvert ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ImpairCarrierInvert?  
[:DigIO:ImpairCarrierInvert ]False

**Related Command(s):** ImpairCarrierInvert

**Description:** Queries the carrier invert status on the Impairment tab of the Digital I/O panel.

## ImpairCarrierInvert

**Command Syntax:** :DigIO:ImpairCarrierInvert Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ImpairCarrierInvert False

**Related Command(s):** ImpairCarrierInvert?

**Description:** Sets the carrier invert status on the Impairment tab of the Digital I/O panel.

## ImpairCommonMode?

**Command Syntax:** :DigIO:ImpairCommonMode?

**Response Syntax:** [:DigIO:ImpairCommonMode ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ImpairCommonMode?  
[:DigIO:ImpairCommonMode ]False

**Related Command(s):** ImpairCommonMode

**Description:** Queries the On/Off status of the Common Mode Sine on the Output Impairment tab of the Digital I/O panel.

## ImpairCommonMode

**Command Syntax:** :DigIO:ImpairCommonMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ImpairCommonMode True

**Related Command(s):** ImpairCommonMode?

**Description:** Turns On/Off the Common Mode Sine on the Output Impairment tab of the Digital I/O panel.

## ImpairCommonModeAmp?

**Command Syntax:** :DigIO:ImpairCommonModeAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ImpairCommonModeAmp ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ImpairCommonModeAmp? Vpp  
[:DigIO:ImpairCommonModeAmp ]1.1

**Related Command(s):** ImpairCommonModeAmp

**Description:** Queries the amplitude of the Common Mode Sine impairment signal.

## ImpairCommonModeAmp

**Command Syntax:** :DigIO:ImpairCommonModeAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairCommonModeAmp 1.1 Vpp

**Related Command(s):** ImpairCommonModeAmp?

**Description:** Sets the amplitude of the Common Mode Sine impairment signal.

## ImpairCommonModeFreq?

**Command Syntax:** :DigIO:ImpairCommonModeFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ImpairCommonModeFreq ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ImpairCommonModeFreq? Hz  
[:DigIO:ImpairCommonModeFreq ]1200

**Related Command(s):** ImpairCommonModeFreq

**Description:** Queries the frequency of the Common Mode Sine impairment signal.

## ImpairCommonModeFreq

**Command Syntax:** :DigIO:ImpairCommonModeFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairCommonModeFreq 4000 Hz

**Related Command(s):** ImpairCommonModeFreq?

**Description:** Sets the frequency of the Common Mode Sine impairment signal.

## ImpairJitterAmp?

**Command Syntax:** :DigIO:ImpairJitterAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ImpairJitterAmp ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ImpairJitterAmp? UI  
[:DigIO:ImpairJitterAmp ]0.012

**Related Command(s):** ImpairJitterAmp

**Description:** Queries the amplitude of the digital audio output impairment jitter signal.

## ImpairJitterAmp

**Command Syntax:** :DigIO:ImpairJitterAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairJitterAmp 0.2 UI

**Related Command(s):** ImpairJitterAmp?

**Description:** Sets the amplitude of the digital audio output impairment jitter signal.

## ImpairJitterFreq?

**Command Syntax:** :DigIO:ImpairJitterFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigIO:ImpairJitterFreq ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ImpairJitterFreq? Hz  
[:DigIO:ImpairJitterFreq ]1000

**Related Command(s):** ImpairJitterFreq

**Description:** Queries the frequency of the digital audio output impairment jitter signal.

## ImpairJitterFreq

**Command Syntax:** :DigIO:ImpairJitterFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairJitterFreq 1000 Hz

**Related Command(s):** ImpairJitterFreq?

**Description:** Sets the frequency of the digital audio output impairment jitter signal.

## ImpairJitterWaveform?

**Command Syntax:** :DigIO:ImpairJitterWaveform?

**Response Syntax:** [:DigIO:ImpairJitterWaveform ]Value

**Response Argument(s):** Value <int> {jtOff=0 | jtSine=1 | jtSquare=2 | jtNoise=3 | jtNoiseBP=4}

**Example:** :DigIO:ImpairJitterWaveform?

[:DigIO:ImpairJitterWaveform ]jtNoise

**Related Command(s):** ImpairJitterWaveform

**Description:** Queries the waveform type of the digital audio output jitter impairment signal.

## ImpairJitterWaveform

**Command Syntax:** :DigIO:ImpairJitterWaveform Value [, AllowCoercion]

**Command Argument(s):** Value <int> {jtOff=0 | jtSine=1 | jtSquare=2 | jtNoise=3 | jtNoiseBP=4 | jtChirp=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ImpairJitterWaveform jtNoise, True

**Related Command(s):** ImpairJitterWaveform?

**Description:** Sets the waveform type for the digital audio output jitter impairment signal.

## SetJitterEq

**Command Syntax:** :DigIO:SetJitterEq FileName

**Command Argument(s):** FileName <string>

**Example:** :DigIO:SetJitterEq FileName

**Description:** Sets the EQ file for the jitter generator.

## ImpairNormalMode?

**Command Syntax:** :DigIO:ImpairNormalMode?

**Response Syntax:** [:DigIO:ImpairNormalMode ]Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigIO:ImpairNormalMode?

[:DigIO:ImpairNormalMode ]False

**Related Command(s):** ImpairNormalMode

**Description:** Queries the on/off status of the digital audio output Normal Mode Noise impairment signal.

## ImpairNormalMode

**Command Syntax:** :DigIO:ImpairNormalMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigIO:ImpairNormalMode True

**Related Command(s):** ImpairNormalMode?

**Description:** Sets the on/off status of the digital audio output Normal Mode Noise impairment signal.

## ImpairNormalModeAmp?

**Command Syntax:** :DigIO:ImpairNormalModeAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitsstring>

**Response Syntax:** [:DigIO:ImpairNormalModeAmp ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigIO:ImpairNormalModeAmp? Vpp  
[:DigIO:ImpairNormalModeAmp ]1.2

**Related Command(s):** ImpairNormalModeAmp

**Description:** Queries the amplitude of the digital audio output Normal Mode Noise impairment signal.

## ImpairNormalModeAmp

**Command Syntax:** :DigIO:ImpairNormalModeAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairNormalModeAmp *Value* [, *AllowCoercion*]

**Related Command(s):** ImpairNormalModeAmp?

## ImpairRiseFallTime?

**Command Syntax:** :DigIO:ImpairRiseFallTime? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitsstring>

**Response Syntax:** [:DigIO:ImpairRiseFallTime ]*Value*

**Example:** :DigIO:ImpairRiseFallTime? [*ValueUnit*]  
[:DigIO:ImpairRiseFallTime ]*Value*

**Related Command(s):** ImpairRiseFallTime

## ImpairRiseFallTime

**Command Syntax:** :DigIO:ImpairRiseFallTime *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigIO:ImpairRiseFallTime *Value* [, *AllowCoercion*]

**Related Command(s):** ImpairRiseFallTime?

## ImpairSendInvalidA?

**Command Syntax:** :DigIO:ImpairSendInvalidA?

**Response Syntax:** [:DigIO:ImpairSendInvalidA ]*Value*

**Response Argument(s):** *Value* <int> {False=0|True=1}

**Example:** :DigIO:ImpairSendInvalidA?  
[:DigIO:ImpairSendInvalidA ]*Value*

**Related Command(s):** ImpairSendInvalidA



## ImpairSendInvalidA

*Command Syntax:* :DigIO:ImpairSendInvalidA Value [, AllowCoercion]

*Command Argument(s):* Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

*Example:* :DigIO:ImpairSendInvalidA Value [, AllowCoercion]

*Related Command(s):* ImpairSendInvalidA?

## ImpairSendInvalidB?

*Command Syntax:* :DigIO:ImpairSendInvalidB?

*Response Syntax:* [:DigIO:ImpairSendInvalidB ]Value

*Response Argument(s):* Value <int> {False=0 | True=1}

*Example:* :DigIO:ImpairSendInvalidB?  
[:DigIO:ImpairSendInvalidB ]Value

*Related Command(s):* ImpairSendInvalidB

## ImpairSendInvalidB

*Command Syntax:* :DigIO:ImpairSendInvalidB Value [, AllowCoercion]

*Command Argument(s):* Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

*Example:* :DigIO:ImpairSendInvalidB Value [, AllowCoercion]

*Related Command(s):* ImpairSendInvalidB?

## Form Commands:

**:DigIO:OpenForm**

**:DigIO:OpenFormwID?**

**:DigIO:CloseForm**

**:DigIO:CloseForms**

**:DigIO:FormCount?**

**:DigIO:FormID?**

## OpenChannelStatusBitsForm

*Command Syntax:* :DigIO:OpenChannelStatusBitsForm

*Command Argument(s):* None

*Example:* :DigIO:OpenChannelStatusBitsForm

*Description:* Opens a channel status bits form on the current page of the page control.

## OpenChannelStatusBitsFormwID?

**Command Syntax:** :DigIO:OpenChannelStatusBitsFormwID?

**Command Argument(s):** None

**Response Syntax:** [:Alyzr(i):Jitter:OpenFormwID] FormID

**Response Argument(s):** FormID <int>

**Example:** :ObjString:OpenFormwID?  
[:ObjString:OpenFormwID] 10

**Description:** Opens a channel status form on the current page of the page control and returns its FormID.

## CloseChannelStatusBitsForm

**Command Syntax:** :DigIO:CloseChannelStatusBitsForm FormID

**Command Argument(s):** FormID <int>

**Example:** :DigIO:CloseChannelStatusBitsForm 10

**Description:** Closes the particular instance of the channel status form with the given FormID.

## CloseChannelStatusBitsForms

**Command Syntax:** :DigIO:CloseChannelStatusBitsForms

**Command Argument(s):** None

**Example:** :DigIO:CloseChannelStatusBitsForms

**Description:** Closes all instances of the form on all pages of the page control.

## ChannelStatusBitsFormCount?

**Command Syntax:** :DigIO:ChannelStatusBitsFormCount?

**Command Argument(s):** None

**Response Syntax:** [:DigIO:ChannelStatusBitsFormCount] Count

**Response Argument(s):** Count <int>

**Example:** :DigIO:ChannelStatusBitsFormCount?  
[:DigIO:ChannelStatusBitsFormCount] 3

**Description:** Returns the number of open channel Status forms on all pages of the page control

## ChannelStatusBitsFormID?

*Command Syntax:* :DigIO:ChannelStatusFormID? *Index*

*Command Argument(s):* *Index* <int>

*Response Syntax:* [:DigIO:ChannelStatusFormID] *FormID*

*Response Argument(s):* *FormID* <int>

*Example:* :DigIO:ChannelStatusFormID? 0  
[:DigIO:ChannelStatusFormID] 10

*Description:* Returns the FormID of the Indexth channel status form. Index=0 corresponds to the first form.

## OpenUserStatusBitsForm

*Command Syntax:* :DigIO:OpenUserStatusBitsForm

*Command Argument(s):* None

*Example:* :DigIO:OpenUserStatusBitsForm

*Description:* Opens a User status bits form on the current page of the page control.

## OpenUserStatusBitsFormwID?

*Command Syntax:* :DigIO:OpenUserStatusBitsFormwID?

*Command Argument(s):* None

*Response Syntax:* [:Alyzr(i):Jitter:OpenFormwID] *FormID*

*Response Argument(s):* *FormID* <int>

*Example:* :ObjString:OpenFormwID?  
[:ObjString:OpenFormwID] 10

*Description:* Opens a User status form on the current page of the page control and returns its FormID.

## CloseUserStatusBitsForm

*Command Syntax:* :DigIO:CloseUserStatusBitsForm *FormID*

*Command Argument(s):* *FormID* <int>

*Example:* :DigIO:CloseUserStatusBitsForm 10

*Description:* Closes the particular instance of the User status form with the given FormID.

## CloseUserStatusBitsForms

*Command Syntax:* :DigIO:CloseUserStatusBitsForms

*Command Argument(s):* None

*Example:* :DigIO:CloseUserStatusBitsForms

*Description:* Closes all instances of the form on all pages of the page control.

## UserStatusBitsFormCount?

*Command Syntax:* :DigIO:UserStatusBitsFormCount?

*Command Argument(s):* None

*Response Syntax:* [:DigIO:UserStatusBitsFormCount] Count

*Response Argument(s):* Count <int>

*Example:* :DigIO:UserStatusBitsFormCount?  
[:DigIO:UserStatusBitsFormCount] 3

*Description:* Returns the number of open User Status forms on all pages of the page control

## UserStatusBitsFormID?

*Command Syntax:* :DigIO:UserStatusFormID? Index

*Command Argument(s):* Index <int>

*Response Syntax:* [:DigIO:UserStatusFormID] FormID

*Response Argument(s):* FormID <int>

*Example:* :DigIO:UserStatusFormID? 0  
[:DigIO:UserStatusFormID] 10

*Description:* Returns the FormID of the Indexth User status form. Index = 0 corresponds to the first form.

## 2.3.6 Sweep

<b>Object:</b>	<b>:Sweep</b>
<b>Object Argument(s):</b>	None
<b>Description:</b>	Commands related to the Sweep Configuration Panel.

### General Sweep Commands

#### Start

**Command Syntax:** :Sweep:Start


**Example:** :Sweep:Start

**Description:** Starts the currently configured sweep.

#### FreeRun

**Command Syntax:** :Sweep:FreeRun

**Example:** :Sweep:FreeRun

**Description:** Starts the "Free Run" mode of SR1 and aborts any sweeps in progress. Equivalent to pressing the  button on the SR1 speed bar.

#### Pause

**Command Syntax:** :Sweep:Pause

**Example:** :Sweep:Pause

**Description:** Pauses either the current sweep or free-run, depending on the current state.

#### pauseResume

**Command Syntax:** :Sweep:pauseResume

**Example:** :Sweep:pauseResume

**Description:** If paused, the command causes sweep or free-run to resume. If in progress, the command pauses the current state.

#### Resume

**Command Syntax:** :Sweep:Resume

**Example:** :Sweep:Resume

**Description:** Resumes either a sweep or free-run that is currently paused.

## GetState

**Command Syntax:** :Sweep:GetState

**Response Syntax:** [:Sweep:GetState ]State

**Response Argument(s):** State <int> {ssFreeRunActive=0 | ssFreeRunPaused=1 | ssSweepActive=2 | ssSweepPaused=3 | ssSweepFinished=4}

**Example:** :Sweep:GetState

**[ :Sweep:GetState ]ssSweepActive**

**Description:** Queries the current state of the Sweep Controller.

## PreSweepDelay?

**Command Syntax:** :Sweep:PreSweepDelay? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Sweep:PreSweepDelay ]Value

**Response Argument(s):** Value <unit>

**Example:** :Sweep:PreSweepDelay? sec

**[ :Sweep:PreSweepDelay ]2.2**

**Related Command(s):** PreSweepDelay

**Description:** Queries the value of the Pre-Sweep Delay.

## PreSweepDelay

**Command Syntax:** :Sweep:PreSweepDelay Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:PreSweepDelay Value [, AllowCoercion]

**Related Command(s):** PreSweepDelay? 2.2 s

**Description:** Sets the value of the Pre-Sweep Delay.

## Repeat?

**Command Syntax:** :Sweep:Repeat?

**Response Syntax:** [:Sweep:Repeat ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Repeat?

**[ :Sweep:Repeat ]False**

**Related Command(s):** Repeat

**Description:** Queries the on/off status of the Sweep Repeat.

## Repeat

**Command Syntax:** `:Sweep:Repeat Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Repeat True`

**Related Command(s):** Repeat?

**Description:** Sets the Sweep Repeat on and off.

## Timeout?

**Command Syntax:** `:Sweep:Timeout? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Timeout ]Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Sweep:Timeout? s`  
`[ :Sweep:Timeout ] 10 s`

**Related Command(s):** Timeout

**Description:** Queries the sweep Timeout value.

## Timeout

**Command Syntax:** `:Sweep:Timeout Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Timeout 2.5 s`

**Related Command(s):** Timeout?

**Description:** Sets the sweep Timeout value.

## Sweep Source Commands

### Sources?

#### Srcs?

**Command Syntax:** `:Sweep:Sources?`

**Response Syntax:** `[:Sweep:Sources ]Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Sweep:Sources?`  
`[ :Sweep:Sources ]1`

**Related Command(s):** Sources

**Description:** Queries the number of configured sweep sources. If no sources are configured the command returns 0. If an inner or outer sweep source is configured the command returns 1. If both sweep axes are configured the command returns 2.

## SourceCfgExternal

**Command Syntax:** :Sweep:SourceCfgExternal *Axis*, *MeasID*

**Command Argument(s):** *Axis* <int> {swpInner=0 | swpOuter=1}

*MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 |  
 msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 |  
 msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 |  
 msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 |  
 msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 |  
 msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
 msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 |  
 msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127  
 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 |  
 msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 |  
 msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 |  
 msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 |  
 msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 |  
 msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 |  
 msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 |  
 msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 |  
 msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 |  
 msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 |  
 msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 |  
 msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 |  
 msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 |  
 msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |  
 msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210  
 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
 msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
 msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
 msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
 msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
 msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
 msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
 msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
 msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
 msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
 msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 |  
 msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
 msA0MThighestToneA=1254 | msA0MThighestToneB=1255 |  
 msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 |  
 msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 |  
 msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 |  
 msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 |  
 msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126  
 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
 msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
 msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
 msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
 msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
 msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
 msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
 msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
 msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 |  
 msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 |



```
msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 |
msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 |
msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 |
msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 | msA1MTspectrumA=2202 |
msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
```

**Example:** :Sweep:SourceCfgExternal swpInner, msA0FFTpectrum

**Description:** Configures the specified sweep axis (inner or outer) as an external based on the measurement referenced by the *MeasID* argument.

## SourceCfgExternalRdg

**Command Syntax:** :Sweep:SourceCfgExternalRdg *Axis*

**Command Argument(s):** *Axis* <int> {swpInner=0 | swpOuter=1}

**Response Syntax:** [:Sweep:SourceCfgExternalRdg ]*MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0lmd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210

```

| msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTdA=1248 |
msA0MTdB=1249 | msA0MTriippleA=1250 | msA0MTriippleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 |
msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 |
msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 |
msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 |
msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 |
msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 |
msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 |
msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 |
msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 |
msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 |
msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 |
msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 | msA1MTspectrumA=2202 |
msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTdA=2248 |
msA1MTdB=2249 | msA1MTriippleA=2250 | msA1MTriippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Sweep:SourceCfgExternalRdg swpInner

**[ :Sweep:SourceCfgExternalRdg ]msAnlgFreqA**

**Description:** Queries the measurement associated with the specified external sweep axis. If the axis is not currently configured as an external sweep the command causes a

Command Error (CME).

## SourceCfgInternal

**Command Syntax:** `:Sweep:SourceCfgInternal Axis, ParamID`

**Command Argument(s):** `Axis <int> {swpInner=0 | swpOuter=1}`  
`ParamID <param>`

**Example:** `:Sweep:SourceCfgInternal swpInner, :AnlgGen:ChA:LoDistSin`

**Description:** Configures the specified sweep axis as an internal sweep based on the parameter specified by the string ParamID. The string used in ParamID is precisely the same string seen in the "Source" window on the Sweep Controller panel when the sweep is configured via the user interface.

## SourceCfgInternalRdg

**Command Syntax:** `:Sweep:SourceCfgInternalRdg Axis`

**Command Argument(s):** `Axis <int> {swpInner=0 | swpOuter=1}`

**Response Syntax:** `[:Sweep:SourceCfgInternalRdg ]ParamID`

**Response Argument(s):** `ParamID <param>`

**Example:** `:Sweep:SourceCfgInternalRdg swpInner`

`[ :Sweep:SourceCfgInternalRdg ] :AnlgGen:ChA:LoDistSine (0)`

**Description:** Queries the parameter string associated with the internal sweep on the specified sweep axis. If the specified axis is not configured as an internal sweep the command triggers a Command Error (CME).

## SourceCfgNone

**Command Syntax:** `:Sweep:SourceCfgNone Axis`

**Command Argument(s):** `Axis <int> {swpInner=0 | swpOuter=1}`

**Example:** `:Sweep:SourceCfgNone swpInner`

**Description:** Removes any sweep source from the specified sweep axis.

## SourceCfgRdg

**Command Syntax:** `:Sweep:SourceCfgRdg Axis`

**Command Argument(s):** `Axis <int> {swpInner=0 | swpOuter=1}`

**Response Syntax:** `[:Sweep:SourceCfgRdg ]Type`

**Response Argument(s):** `Type <int> {estNone=0 | estInternal=1 | estExternal=2 | estTime=3 | estSwitcher=4}`

**Example:** `:Sweep:SourceCfgRdg swpInner`

`[ :Sweep:SourceCfgRdg ]estInternal`

**Description:** Queries the type of sweep source connected to the specified axis.

## SourceCfgSwitcher

**Command Syntax:** `:Sweep:SourceCfgSwitcher Axis`

**Command Argument(s):** `Axis <int> {swpInner=0 | swpOuter=1}`

**Example:** `:Sweep:SourceCfgSwitcher swpInner`

**Description:** Sets the sweep source for the specified axis to a Switcher Sweep.

## SourceCfgTime

**Command Syntax:** :Sweep:SourceCfgTime *Axis*, *TimeID*

**Command Argument(s):** *Axis* <int> {swpInner=0 | swpOuter=1}  
*TimeID* <int> {ttIntersampleDelay=0 | ttCorrectedTime=1}

**Example:** :Sweep:SourceCfgTime swpInner, ttIntersampleDelay

**Description:** Sets the sweep source for the specified axis to a Time Sweep using the specified time mode.

## SourceCfgTimeRdg

**Command Syntax:** :Sweep:SourceCfgTimeRdg *Axis*

**Command Argument(s):** *Axis* <int> {swpInner=0 | swpOuter=1}

**Response Syntax:** [:Sweep:SourceCfgTimeRdg ]*TimeID*

**Response Argument(s):** *TimeID* <int> {ttIntersampleDelay=0 | ttCorrectedTime=1}

**Example:** :Sweep:SourceCfgTimeRdg swpInner  
 [:Sweep:SourceCfgTimeRdg ]ttCorectedTime

**Description:** Queries the time mode associated with the time sweep source on the specified axis. If the specified axis is not configured as a time sweep, a Command Error (CME) results.

## Sweep Data Commands

### DataCfg

**Command Syntax:** :Sweep:DataCfg *DataID*, *MeasID*

**Command Argument(s):** *DataID* <int>

*MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 |  
 msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 |  
 msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 |  
 msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 |  
 msA0FFT2linSpec=1112 | msA0FFT2linPhase=1113 | msA0FFT2timeRecA=1120 |  
 msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
 msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 |  
 msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 |  
 msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 |  
 msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 |  
 msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 |  
 msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 |  
 msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 |  
 msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 |  
 msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 |  
 msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 |  
 msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 |  
 msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 |  
 msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 |  
 msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 |  
 msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 |  
 msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 |

```

msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210
| msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 |
msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 |
msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 |
msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 |
msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 |
msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126
| msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 |
msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 |
msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 |
msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 |
msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 |
msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 | msA1MTspectrumA=2202 |
msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210
| msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Sweep:DataCfg 0, msAnlgFreqA

**Description:** Sets the measurement for the sweep data slot referenced by *DataID* to the measurement specified by the argument *MeasID*. The measurement codes for all possible SR1 measurements on each analyzer (A0 and A1) are given in the table

of enumerations above.

## DataCfgRdg

**Command Syntax:** :Sweep:DataCfgRdg *DataID*

**Command Argument(s):** *DataID* <int>

**Response Syntax:** [:Sweep:DataCfgRdg ]*MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |

```

msA 1jitFreqDomLinPhase=2173 | msA 1jitFreqDomJitter=2174 |
msA 1jitTimeDomJitter=2175 | msA 1jitPhysSampRate=2176 |
msA 1HistoTimeRecA=2180 | msA 1HistoTimeRecB=2181 |
msA 1HistoHistoA=2182 | msA 1HistoHistoB=2183 | msA 1HistoProbA=2184 |
msA 1HistoProbB=2185 | msA 1HistoFitA=2186 | msA 1HistoFitB=2187 |
msA 1HistoFitMeanA=2188 | msA 1HistoFitMeanB=2189 |
msA 1HistoFitSigmaA=2190 | msA 1HistoFitSigmaB=2191 |
msA 1MTtimeRecA=2200 | msA 1MTtimeRecB=2201 | msA 1MTspectrumA=2202 |
msA 1MTspectrumB=2203 | msA 1MTthdnBinsA=2204 |
msA 1MTthdnBinsB=2205 | msA 1MTThdBinsA=2206 | msA 1MTThdBinsB=2207 |
msA 1MTimdBinsA=2208 | msA 1MTimdBinsB=2209 | msA 1MTnoiseBinsA=2210 |
msA 1MTnoiseBinsB=2211 | msA 1MTtdBinsA=2212 | msA 1MTtdBinsB=2213 |
msA 1MTfreqRespMagA=2220 | msA 1MTfreqRespMagB=2221 |
msA 1MTfreqRespPhaseA=2222 | msA 1MTfreqRespPhaseB=2223 |
msA 1MTthdnVsFreqA=2224 | msA 1MTthdnVsFreqB=2225 |
msA 1MTthdVsFreqA=2226 | msA 1MTthdVsFreqB=2227 |
msA 1MTimdVsFreqA=2228 | msA 1MTimdVsFreqB=2229 |
msA 1MTxtalkVsFreqAB=2230 | msA 1MTxtalkVsFreqBA=2231 |
msA 1MTthdnA=2240 | msA 1MTthdnB=2241 | msA 1MTthdA=2242 |
msA 1MTthdB=2243 | msA 1MTimdA=2244 | msA 1MTimdB=2245 |
msA 1MTnoiseA=2246 | msA 1MTnoiseB=2247 | msA 1MTtdA=2248 |
msA 1MTtdB=2249 | msA 1MTripleA=2250 | msA 1MTripleB=2251 |
msA 1MTlowestToneA=2252 | msA 1MTlowestToneB=2253 |
msA 1MThighestToneA=2254 | msA 1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Sweep:DataCfgRdg 0  
[:Sweep:DataCfgRdg ]msA1FFTlinSpec

**Description:** Queries the measurement for the sweep data slot referenced by *DataID* to the measurement specified by the argument *MeasID*. The measurement codes for all possible SRI measurements on each analyzer (A0 and A1) are given in the table of enumerations above.

## External Sweep Commands

### StartMethod?

**Command Syntax:** :Sweep:StartMethod?

**Response Syntax:** [:Sweep:StartMethod ]*Value*

**Response Argument(s):** *Value* <int> {ssStartWithinTo=0 | ssStartWithinRange=1 | ssStartAnywhere=2}

**Example:** :Sweep:StartMethod?  
[:Sweep:StartMethod ]ssStartWithinRange

**Related Command(s):** StartMethod

**Description:** Queries the external sweep "Start Sweep At" selection.

## StartMethod

**Command Syntax:** :Sweep:StartMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ssStartWithinTol=0 | ssStartWithinRange=1 | ssStartAnywhere=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:StartMethod ssStartWithinTol

**Related Command(s):** StartMethod?

**Description:** Sets the external sweep "Start Sweep At" selection.

## StopMethod?

**Command Syntax:** :Sweep:StopMethod?

**Response Syntax:** [:Sweep:StopMethod ]*Value*

**Response Argument(s):** *Value* <int> {ssStopWithinTol=0 | ssStopOnReturn=1}

**Example:** :Sweep:StopMethod?

**[ :Sweep:StopMethod ] ssStopOnReturn**

**Related Command(s):** StopMethod

**Description:** Queries the external sweep "Stop Sweep At" selection.

## StopMethod

**Command Syntax:** :Sweep:StopMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ssStopWithinTol=0 | ssStopOnReturn=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:StopMethod *Value* [, *AllowCoercion*]

**Related Command(s):** StopMethod?

**Description:** Sets the external sweep "Stop Sweep At" selection.

## ExtMinLevelCfg

**Command Syntax:** :Sweep:ExtMinLevelCfg *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 |  
msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 |  
msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 |  
msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 |  
msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 |  
msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 |  
msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 |  
msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 |  
msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 |  
msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 |  
msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0lmd=1160 |  
msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 |  
msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 |  
msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 |  
msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 |  
msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 |  
msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 |  
msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 |  
msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 |



```

msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 |
msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |
msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |
msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210
| msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 |
msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 |
msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 |
msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 |
msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 |
msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126
| msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 |
msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 |
msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 |
msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 |
msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 |
msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 | msA1MTspectrumA=2202 |
msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210
| msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Sweep:ExtMinLevelCfg msAnlgFreqA

**Description:** Sets the Measurement associated with the External Sweep Minimum Level. If

configured, the external sweep waits for the specified measurement to reach the set minimum value before looking for a new x-axis point.

## ExtMinLevelCfgRdg

**Command Syntax:** :Sweep:ExtMinLevelCfgRdg

**Response Syntax:** [:Sweep:ExtMinLevelCfgRdg ]*MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqA=1230 | msA0MTxtalkVsFreqB=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTtripleA=1250 | msA0MTtripleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |

```

msA 1jitTimeDomJitter=2175 | msA 1jitPhysSampRate=2176 |
msA 1HistoTimeRecA=2180 | msA 1HistoTimeRecB=2181 |
msA 1HistoHistoA=2182 | msA 1HistoHistoB=2183 | msA 1HistoProbA=2184 |
msA 1HistoProbB=2185 | msA 1HistoFitA=2186 | msA 1HistoFitB=2187 |
msA 1HistoFitMeanA=2188 | msA 1HistoFitMeanB=2189 |
msA 1HistoFitSigmaA=2190 | msA 1HistoFitSigmaB=2191 |
msA 1MTtimeRecA=2200 | msA 1MTtimeRecB=2201 | msA 1MTspectrumA=2202 |
msA 1MTspectrumB=2203 | msA 1MTthdnBinsA=2204 |
msA 1MTthdnBinsB=2205 | msA 1MTthdBinsA=2206 | msA 1MTthdBinsB=2207 |
msA 1MTimdBinsA=2208 | msA 1MTimdBinsB=2209 | msA 1MTnoiseBinsA=2210 |
msA 1MTnoiseBinsB=2211 | msA 1MTtdBinsA=2212 | msA 1MTtdBinsB=2213 |
msA 1MTfreqRespMagA=2220 | msA 1MTfreqRespMagB=2221 |
msA 1MTfreqRespPhaseA=2222 | msA 1MTfreqRespPhaseB=2223 |
msA 1MTthdnVsFreqA=2224 | msA 1MTthdnVsFreqB=2225 |
msA 1MTthdVsFreqA=2226 | msA 1MTthdVsFreqB=2227 |
msA 1MTimdVsFreqA=2228 | msA 1MTimdVsFreqB=2229 |
msA 1MTxtalkVsFreqAB=2230 | msA 1MTxtalkVsFreqBA=2231 |
msA 1MTthdnA=2240 | msA 1MTthdnB=2241 | msA 1MTthdA=2242 |
msA 1MTthdB=2243 | msA 1MTimdA=2244 | msA 1MTimdB=2245 |
msA 1MTnoiseA=2246 | msA 1MTnoiseB=2247 | msA 1MTtdA=2248 |
msA 1MTtdB=2249 | msA 1MTripleA=2250 | msA 1MTripleB=2251 |
msA 1MTlowestToneA=2252 | msA 1MTlowestToneB=2253 |
msA 1MThighestToneA=2254 | msA 1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** `:Sweep:ExtMinLevelCfgRdg`  
`[ :Sweep:ExtMinLevelCfgRdg ]msNull`

**Description** Queries the Measurement associated with the External Sweep Minimum Level. If configured, the external sweep waits for the specified measurement to reach the set minimum value before looking for a new x-axis point

## Supported Form Commands:

- :Sweep:OpenForm**
- :Sweep:OpenFormwID?**
- :Sweep:CloseForm**
- :Sweep:CloseForms**
- :Sweep:FormCount?**
- :Sweep:FormID?**

### **OpenSettlingForm**

**Command Syntax:** `:Sweep:OpenSettlingForm`

**Command Argument(s):** None

**Example:** `:Sweep:OpenSettlingForm`

**Description:** Opens a settling form on the current page of the page control.

## OpenSettlingFormwID?

**Command Syntax:** :Sweep:OpenSettlingFormwID?

**Command Argument(s):** None

**Response Syntax:** [:Sweep:OpenSettlingFormwID] *FormID*

**Response Argument(s):** *FormID* <int>

**Example:** :Sweep:OpenSettlingFormwID?  
[:Sweep:OpenSettlingFormwID] 10

**Description:** Opens a settling form on the current page of the page control and returns its FormID.

## CloseSettlingForm

**Command Syntax:** :Sweep:CloseSettlingForm *FormID*

**Command Argument(s):** *FormID* <int>

**Example:** :Sweep:CloseSettlingForm 10

**Description:** Closes the particular instance of the settling form with the given FormID.

## CloseSettlingForms

**Command Syntax:** :Sweep:CloseSettlingForms

**Command Argument(s):** None

**Example:** :Sweep:CloseSettlingForms

**Description:** Closes all instances of the form on all pages of the page control.

## SettlingFormCount?

**Command Syntax:** :Sweep:SettlingFormCount?

**Command Argument(s):** None

**Response Syntax:** [:Sweep:SettlingFormCount] *Count*

**Response Argument(s):** *Count* <int>

**Example:** :Sweep:SettlingFormCount?  
[:Sweep:SettlingFormCount] 3

**Description:** Returns the number of open settling forms on all pages of the page control

## SettlingFormID?

**Command Syntax:** :Sweep:SettlingFormID? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:Sweep:SettlingFormID] *FormID*

**Response Argument(s):** *FormID* <int>

**Example:** :Sweep:SettlingFormID? 0  
[:Sweep:SettlingFormID] 10

**Description:** Returns the FormID of the Index<sup>th</sup> settling form. Index = 0 corresponds to the first form.

2.3.6.1 Sweep Source

<b>Object:</b>	<b>:Sweep:Source(<i>Axis</i>)</b>
<b>Object Argument(s):</b>	<i>Axis</i> <int> {swpInner=0   swpOuter=1}
<b>Description:</b>	Commands related to the configuration of the inner or outer sweep axis.

## Internal Sweep Source Commands

### InternLinStep?

**Command Syntax:** :Sweep:Source(*I*):InternLinStep? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Source(*I*):InternLinStep ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Source(0):InternLinStep? Hz  
 [:Sweep:Source(0):InternLinStep 1000

**Related Command(s):** InternLinStep

**Description:** Queries the value of linear sweep step for the internal sweep source associated with the specified axis.

### InternLinStep

**Command Syntax:** :Sweep:Source(*I*):InternLinStep *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:Source(0):InternLinStep 1000 Hz

**Related Command(s):** InternLinStep?

**Description:** Sets the value of linear sweep step for the internal sweep source associated with the specified axis.

### InternLogStep?

**Command Syntax:** :Sweep:Source(*I*):InternLogStep? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Source(*I*):InternLogStep ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Source(0):InternLogStep?  
 [:Sweep:Source(0):InternLogStep ] 1.1

**Related Command(s):** InternLogStep

**Description:** Queries the value of the log sweep step for the internal sweep source associated with the specified axis.

## InternLogStep

**Command Syntax:** `:Sweep:Source(I):InternLogStep Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(0):InternLogStep 1.1`

**Related Command(s):** InternLogStep?

**Description:** Sets the value of the log sweep step for the internal sweep source associated with the specified axis.

## IntNumSteps?

**Command Syntax:** `:Sweep:Source(I):IntNumSteps?`

**Response Syntax:** `[:Sweep:Source(I):IntNumSteps ]Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Sweep:Source(0):IntNumSteps?`  
`[ :Sweep:Source(0):IntNumSteps ]100`

**Related Command(s):** Queries the number of steps of the internal sweep source associated with the specified axis.

## IntNumSteps

**Command Syntax:** `:Sweep:Source(I):IntNumSteps Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(0):IntNumSteps 100`

**Related Command(s):** IntNumSteps?

**Description:** Sets the number of steps of the internal sweep source associated with the specified axis.

## Log?

**Command Syntax:** `:Sweep:Source(I):Log?`

**Response Syntax:** `[:Sweep:Source(I):Log ]Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Sweep:Source(1):Log?`  
`[ :Sweep:Source(1):Log ] True`

**Related Command(s):** Log

**Description:** Queries whether the internal sweep source associated with the specified axis is in log sweep mode.

## Log

**Command Syntax:** `:Sweep:Source(I):Log Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(1):Log True`

**Related Command(s):** Log?

**Description:** Sets whether the internal sweep source associated with the specified axis is in log sweep mode.

## TableName?

**Command Syntax:** `:Sweep:Source(I):TableName?`

**Response Syntax:** `[:Sweep:Source(I):TableName]Value`

**Response Argument(s):** `Value <string>`

**Example:** `:Sweep:Source(1):TableName?`  
`[ :Sweep:Source(1):TableName ] MyTable.txt`

**Related Command(s):** TableName

**Description:** Queries the table filename used in conjunction with table-driven internal sweeps.

## TableName

**Command Syntax:** `:Sweep:Source(I):TableName Value [, AllowCoercion]`

**Command Argument(s):** `Value <string>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(0):TableName MyTable.txt`

**Related Command(s):** TableName?

**Description:** Sets the table filename used in conjunction with table-driven internal sweeps. The default directory for table sweep files is the "user" directory. Thus, filenames specified without any additional path information will be expected to be found in the "user" directory.

## TableColIndex?

**Command Syntax:** `:Sweep:Source(I):TableColIndex?`

**Response Syntax:** `[:Sweep:Source(I):TableColIndex]Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Sweep:Source(0):TableColIndex?`  
`[ :Sweep:Source(0):TableColIndex ] 2`

**Related Command(s):** TableColIndex

**Description:** Queries the index of the column that will be used to obtain table-sweep data. The first column has index 1, the second, 2, etc.

## TableColIndex

**Command Syntax:** :Sweep:Source(I):TableColIndex *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:Source(0):TableColIndex 2

**Related Command(s):** TableColIndex?

**Description:** Sets the index of the column in the table sweep file that will be used to obtain table-sweep data. The first column has index 1, the second, 2, etc.

## LoadSweepTable

**Command Syntax:** :Sweep:Source(I):LoadSweepTable *FileName*, *ColIndex*

**Command Argument(s):** *FileName* <string>  
*ColIndex* <int>

**Example:** :Sweep:Source(1):LoadSweepTable MyFile.txt 2

**Related Command(s):** Loads Sweep Table data from the specified column of the named file.

## External Sweep Source Commands

Note that although it is still necessary to specify the sweep axis index when using these commands, external sweeps are only allowed on the inner sweep axis.

### ExternStep?

**Command Syntax:** :Sweep:Source(I):ExternStep? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Source(I):ExternStep ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Source(0):ExternStep? vrms  
[:Sweep:Source(0):ExternStep ] 0.01

**Related Command(s):** ExternStep

**Description:** Queries the value of the external sweep step for the inner sweep axis.

### ExternStep

**Command Syntax:** :Sweep:Source(I):ExternStep *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:Source(0):ExternStep 0.01 vrms

**Related Command(s):** ExternStep?

**Description:** Sets the value of the external sweep step for the inner sweep axis. This command sets the value of the step size in absolute units, for instance .1 Vrms. To set the value of the step size in a relative unit such as percent or db use the ExternStepRel command.



## ExternStepRel?

**Command Syntax:** `:Sweep:Source(I):ExternStepRel? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Source(I):ExternStepRel ]Value`

**Example:** `:Sweep:Source(0):ExternStepRel? %  
[:Sweep:Source(0):ExternStepRel ] 4 %`

**Related Command(s):** `ExternStepRel`

**Description:** Queries the value of the external relative sweep step for the inner sweep axis.

## ExternStepRel

**Command Syntax:** `:Sweep:Source(I):ExternStepRel Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>  
AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(0):ExternStepRel 4 %`

**Related Command(s):** `ExternStepRel?`

**Description:** Sets the value of the external relative sweep step for the inner sweep axis.

## UseExternStepRel?

**Command Syntax:** `:Sweep:Source(I):UseExternStepRel?`

**Response Syntax:** `[:Sweep:Source(I):UseExternStepRel ]Value`

**Response Argument(s):** `Value <int> {False=0|True=1}`

**Example:** `:Sweep:Source(0):UseExternStepRel?  
[:Sweep:Source(0):UseExternStepRel ] True`

**Related Command(s):** `UseExternStepRel`

**Description:** Queries whether the external sweep will used the relative step-size (true) or the absolute step size (false).

## UseExternStepRel

**Command Syntax:** `:Sweep:Source(I):UseExternStepRel Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0|True=1}  
AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(0):UseExternStepRel True`

**Related Command(s):** `UseExternStepRel?`

**Description:** Sets whether the external sweep will used the relative step-size (true) or the absolute step size (false).

## MinLevel?

**Command Syntax:** :Sweep:Source(I):MinLevel? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Source(I):MinLevel ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Source(0):MinLevel? Vrms  
[:Sweep:Source(0):MinLevel ]0.001 Vrms

**Related Command(s):** MinLevel

**Description:** Queries the value of the minimum level parameter for external sweeps. The specified minimum level measurement must exceed the minimum level before a valid sweep point is recorded.

## MinLevel

**Command Syntax:** :Sweep:Source(I):MinLevel *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Sweep:Source(0):MinLevel 0.001 Vrms

**Related Command(s):** MinLevel?

**Description:** Sets the value of the minimum level parameter for external sweeps. The specified minimum level measurement must exceed the minimum level before a valid sweep point is recorded.

## StartTol?

**Command Syntax:** :Sweep:Source(I):StartTol? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Source(I):StartTol ]*Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Source(0):StartTol? Hz  
[:Sweep:Source(0):StartTol ]10 Hz

**Related Command(s):** StartTol

**Description:** Queries the value of the absolute start tolerance. The measured external sweep parameter must be within start tolerance of the Start Value for the external sweep to start. This command queries the absolute start tolerance, e.g. 10 Hz. To query or set the relative start tolerance (e.g. 10%) use the StartTolRel command.

## StartTol

**Command Syntax:** :Sweep:Source(I):StartTol Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Sweep:Source(0):StartTol 10 Hz

**Related Command(s):** StartTol?

**Description:** Sets the value of the absolute start tolerance. The measured external sweep parameter must be within start tolerance of the Start Value for the external sweep to start. This command queries the absolute start tolerance, e.g. 10 Hz. To query or set the relative start tolerance (e.g. 10%) use the StartTolRel command

## StartTolRel?

**Command Syntax:** :Sweep:Source(I):StartTolRel? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Sweep:Source(I):StartTolRel ]Value

**Response Argument(s):** Value <unit>

**Example:** :Sweep:Source(0):StartTolRel? %  
[:Sweep:Source(0):StartTolRel ] 10 %

**Related Command(s):** StartTolRel

**Description:** Queries the value of the relative start tolerance. The measured external sweep parameter must be within the start tolerance for the external sweep to start. This command queries the relative start tolerance, e.g. 10%. To query or set the relative start tolerance (e.g. 10 Hz) use the StartTol command.

## StartTolRel

**Command Syntax:** :Sweep:Source(I):StartTolRel Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Sweep:Source(0):StartTolRel 10 %

**Related Command(s):** StartTolRel?

**Description:** Sets the value of the relative start tolerance. The measured external sweep parameter must be within the start tolerance for the external sweep to start. This command queries the relative start tolerance, e.g. 10%. To query or set the relative start tolerance (e.g. 10 Hz) use the StartTol command.

## UseStartTolRel?

**Command Syntax:** :Sweep:Source(I):UseStartTolRel?

**Response Syntax:** [:Sweep:Source(I):UseStartTolRel ]Value

**Response Argument(s):** Value <int> {False=0|True=1}

**Example:** :Sweep:Source(0):UseStartTolRel?  
[:Sweep:Source(0):UseStartTolRel ]Value

**Related Command(s):** UseStartTolRel

**Description:** Queries whether the relative (true) or absolute (false) start tolerance is used when deciding to start an external sweep.

## UseStartTolRel

**Command Syntax:** `:Sweep:Source(I):UseStartTolRel Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(0):UseStartTolRel Value [, AllowCoercion]`

**Related Command(s):** UseStartTolRel?

**Description:** Sets whether the relative (true) or absolute (false) start tolerance is used when deciding to start an external sweep.

## StartVal?

**Command Syntax:** `:Sweep:Source(I):StartVal? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Source(I):StartVal]Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Sweep:Source(0):StartVal? Hz`  
`[:Sweep:Source(0):StartVal]20 Hz`

**Related Command(s):** StartVal

**Description:** Queries the desired starting value for the external sweep.

## StartVal

**Command Syntax:** `:Sweep:Source(I):StartVal Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Source(0):StartVal 20 Hz`

**Related Command(s):** StartVal?

**Description:** Sets the desired starting value for the external sweep.

## StopTol?

**Command Syntax:** `:Sweep:Source(I):StopTol? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Source(I):StopTol]Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Sweep:Source(0):StopTol? Hz`  
`[:Sweep:Source(0):StopTol]100 Hz`

**Related Command(s):** StopTol

**Description:** Queries the value of the absolute stop tolerance. The measured external sweep parameter must be within stop tolerance of the Stop Value for the external sweep to stop. This command queries the absolute stop tolerance, e.g. 10 Hz. To query or set the relative stop tolerance (e.g. 10 %) use the StopTolRel command.

## StopTol

**Command Syntax:** `:Sweep:Source(I):StopTol Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(0):StopTol 100 Hz`

**Related Command(s):** StopTol?

**Description:** Sets the value of the absolute stop tolerance. The measured external sweep parameter must be within stop tolerance of the Stop Value for the external sweep to stop. This command queries the absolute stop tolerance, e.g. 10 Hz. To query or set the relative stop tolerance (e.g. 10 %) use the StopTolRel command.

## StopTolRel?

**Command Syntax:** `:Sweep:Source(I):StopTolRel? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Source(I):StopTolRel]Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Sweep:Source(0):StopTolRel? pct`  
`[ :Sweep:Source(0):StopTolRel ] 10 pct`

**Related Command(s):** StopTolRel

**Description:** Queries the value of the relative stop tolerance. The measured external sweep parameter must be within stop tolerance of the Stop Value for the external sweep to stop. This command queries the relative stop tolerance, e.g. 10 %. To query or set the absolute stop tolerance (e.g. 10 Hz) use the StopTol command.

## StopTolRel

**Command Syntax:** `:Sweep:Source(I):StopTolRel Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(0):StopTolRel 10 pct, true`

**Related Command(s):** StopTolRel?

**Description:** Sets the value of the relative stop tolerance. The measured external sweep parameter must be within stop tolerance of the Stop Value for the external sweep to stop. This command queries the relative stop tolerance, e.g. 10 %. To query or set the absolute stop tolerance (e.g. 10 Hz) use the StopTol command.

## UseStopTolRel?

**Command Syntax:** `:Sweep:Source(I):UseStopTolRel?`

**Response Syntax:** `[:Sweep:Source(I):UseStopTolRel]Value`

**Response Argument(s):** `Value <int> {False=0|True=1}`

**Example:** `:Sweep:Source(0):UseStopTolRel?`  
`[ :Sweep:Source(0):UseStopTolRel ] false`

**Related Command(s):** UseStopTolRel

**Description:** Queries whether the relative (true) or absolute (false) stop tolerance is used when deciding to stop an external sweep.

## UseStopTolRel

**Command Syntax:** :Sweep:Source(I):UseStopTolRel Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(0):UseStopTolRel false

**Related Command(s):** UseStopTolRel?

**Description:** Sets whether the relative (true) or absolute (false) stop tolerance is used when deciding to stop an external sweep.

## StopVal?

**Command Syntax:** :Sweep:Source(I):StopVal? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Sweep:Source(I):StopVal ]Value

**Response Argument(s):** Value <unit>

**Example:** :Sweep:Source(0):StopVal? [ValueUnit]  
[:Sweep:Source(0):StopVal ]Value

**Related Command(s):** StopVal

**Description:** Queries the desired stop value for the external sweep axis.

## StopVal

**Command Syntax:** :Sweep:Source(I):StopVal Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

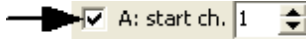
**Example:** :Sweep:Source(0):StopVal Value [, AllowCoercion]

**Related Command(s):** StopVal?

**Description:** Sets the desired stop value for the external sweep axis.

## Switcher Sweep Commands

Note that although it is still necessary to specify the sweep axis index when using these commands, external sweeps are only allowed on the outer sweep axis.



### InSwitchEnableChA?

**Command Syntax:** :Sweep:Source(I):InSwitchEnableChA?

**Response Syntax:** [:Sweep:Source(I):InSwitchEnableChA ]Value

**Response Argument(s):** Value <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchEnableChA?  
[:Sweep:Source(1):InSwitchEnableChA ]True

**Related Command(s):** InSwitchEnableChA

**Description:** Queries the enable status of the Input Switch channel A sweep.

### InSwitchEnableChA

**Command Syntax:** :Sweep:Source(I):InSwitchEnableChA Value [, AllowCoercion]

**Command Argument(s):** Value <bool> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchEnableChA True

**Related Command(s):** InSwitchEnableChA?

**Description:** Sets the enable status of the Input Switch channel A sweep.

### InSwitchEnableChB?

**Command Syntax:** :Sweep:Source(I):InSwitchEnableChB?

**Response Syntax:** [:Sweep:Source(I):InSwitchEnableChB ]Value

**Response Argument(s):** Value <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchEnableChB?  
[:Sweep:Source(1):InSwitchEnableChB ]Value

**Related Command(s):** InSwitchEnableChB

**Description:** Queries the enable status of the Input Switch channel B sweep.

### InSwitchEnableChB

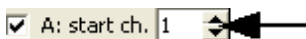
**Command Syntax:** :Sweep:Source(I):InSwitchEnableChB Value [, AllowCoercion]

**Command Argument(s):** Value <bool> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchEnableChB Value [, AllowCoercion]

**Related Command(s):** InSwitchEnableChB?

**Description:** Sets the enable status of the Input Switch channel B sweep.



## InSwitchStartChA?

**Command Syntax:** :Sweep:Source(I):InSwitchStartChA?

**Response Syntax:** [:Sweep:Source(I):InSwitchStartChA ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):InSwitchStartChA?  
[:Sweep:Source(1):InSwitchStartChA ]Value

**Related Command(s):** InSwitchStartChA

**Description:** Queries the starting logical channel number for the channel A input switcher sweep.

## InSwitchStartChA

**Command Syntax:** :Sweep:Source(I):InSwitchStartChA Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Sweep:Source(1):InSwitchStartChA Value [,  
AllowCoercion]

**Related Command(s):** InSwitchStartChA?

**Description:** Sets the starting logical channel number for the channel A input switcher sweep.

## InSwitchStartChB?

**Command Syntax:** :Sweep:Source(I):InSwitchStartChB?

**Response Syntax:** [:Sweep:Source(I):InSwitchStartChB ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):InSwitchStartChB?  
[:Sweep:Source(1):InSwitchStartChB ]Value

**Related Command(s):** InSwitchStartChB

**Description:** Queries the starting logical channel number for the channel B input switcher sweep.

## InSwitchStartChB

**Command Syntax:** :Sweep:Source(I):InSwitchStartChB Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Sweep:Source(I):InSwitchStartChB Value [,  
AllowCoercion]

**Related Command(s):** InSwitchStartChB?

**Description:** Sets the starting logical channel number for the channel B input switcher sweep.



## InSwitchName?

**Command Syntax:** :Sweep:Source(I):InSwitchName?

**Response Syntax:** [:Sweep:Source(I):InSwitchName ]Value

**Response Argument(s):** Value <string>

**Example:** :Sweep:Source(1):InSwitchName?  
[:Sweep:Source(1):InSwitchName ]BNC

**Related Command(s):** InSwitchName

**Description:** Queries the network name for the input sweep

## InSwitchName

**Command Syntax:** :Sweep:Source(I):InSwitchName Value [, AllowCoercion]

**Command Argument(s):** Value <string>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchName BNC

**Related Command(s):** InSwitchName?

**Description:** Sets the network name for the input sweep

in steps of

## InSwitchStep?

**Command Syntax:** :Sweep:Source(I):InSwitchStep?

**Response Syntax:** [:Sweep:Source(I):InSwitchStep ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):InSwitchStep?  
[:Sweep:Source(1):InSwitchStep ]Value

**Related Command(s):** InSwitchStep

**Description:** Queries the logical channel number step size for the input switcher sweep.

## InSwitchStep

**Command Syntax:** :Sweep:Source(I):InSwitchStep Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):InSwitchStep Value [, AllowCoercion]

**Related Command(s):** InSwitchStep?

**Description:** Sets the logical channel number step size for the input switcher sweep.

Bus

## OutSwitchBusChA?

**Command Syntax:** :Sweep:Source(I):OutSwitchBusChA?

**Response Syntax:** [:Sweep:Source(I):OutSwitchBusChA ]Value

**Response Argument(s):** Value <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchBusChA?  
[:Sweep:Source(1):OutSwitchBusChA ]True

**Related Command(s):** OutSwitchBusChA

**Description:** Queries the bus status for the A channel output switch sweep. When bus is enabled all the output channels will be connected to the SRS source except one, which will be swept according to the specified parameters.

## OutSwitchBusChA

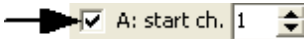
**Command Syntax:** :Sweep:Source(I):OutSwitchBusChA Value [, AllowCoercion]

**Command Argument(s):** Value <bool> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchBusChA True

**Related Command(s):** OutSwitchBusChA?

**Description:** Queries the bus status for the A channel output switch sweep. When bus is enabled all the output channels will be connected to the SRS source except one, which will be swept according to the specified parameters.



## OutSwitchEnableChA?

**Command Syntax:** :Sweep:Source(I):OutSwitchEnableChA?

**Response Syntax:** [:Sweep:Source(I):OutSwitchEnableChA ]Value

**Response Argument(s):** Value <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchEnableChA?  
[:Sweep:Source(1):OutSwitchEnableChA ]False

**Related Command(s):** OutSwitchEnableChA

**Description:** Queries the enable status of the Output Switch channel A sweep.

## OutSwitchEnableChA

**Command Syntax:** :Sweep:Source(I):OutSwitchEnableChA Value [, AllowCoercion]

**Command Argument(s):** Value <bool> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchEnableChA False]

**Related Command(s):** OutSwitchEnableChA?

**Description:** Sets the enable status of the Output Switch channel A sweep.

## OutSwitchEnableChB?

**Command Syntax:** :Sweep:Source(I):OutSwitchEnableChB?

**Response Syntax:** [:Sweep:Source(I):OutSwitchEnableChB ]Value

**Response Argument(s):** Value <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchEnableChB?

**[ :Sweep:Source(1):OutSwitchEnableChB ]True**

**Related Command(s):** OutSwitchEnableChB

**Description:** Queries the enable status of the Output Switch channel B sweep.

## OutSwitchEnableChB

**Command Syntax:** :Sweep:Source(I):OutSwitchEnableChB Value [, AllowCoercion]

**Command Argument(s):** Value <bool> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchEnableChB True

**Related Command(s):** OutSwitchEnableChB?

**Description:** Sets the enable status of the Output Switch channel B sweep.

## OutSwitchName?

**Command Syntax:** :Sweep:Source(I):OutSwitchName?

**Response Syntax:** [:Sweep:Source(I):OutSwitchName ]Value

**Response Argument(s):** Value <string>

**Example:** :Sweep:Source(1):OutSwitchName?

**[ :Sweep:Source(1):OutSwitchName ]Value**

**Related Command(s):** OutSwitchName

## OutSwitchName

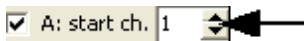
**Command Syntax:** :Sweep:Source(I):OutSwitchName Value [, AllowCoercion]

**Command Argument(s):** Value <string>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(I):OutSwitchName Value [, AllowCoercion]

**Related Command(s):** OutSwitchName?



## OutSwitchStartChA?

**Command Syntax:** :Sweep:Source(I):OutSwitchStartChA?

**Response Syntax:** [:Sweep:Source(I):OutSwitchStartChA ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):OutSwitchStartChA?

**[ :Sweep:Source(1):OutSwitchStartChA ]Value**

**Related Command(s):** OutSwitchStartChA

**Description:** Queries the starting logical channel number for the A-channel output switcher sweep.

## OutSwitchStartChA

**Command Syntax:** :Sweep:Source(I):OutSwitchStartChA Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchStartChA Value [, AllowCoercion]

**Related Command(s):** OutSwitchStartChA?

**Description:** Sets the starting logical channel number for the A-channel output switcher sweep.

## OutSwitchStartChB?

**Command Syntax:** :Sweep:Source(I):OutSwitchStartChB?

**Response Syntax:** [:Sweep:Source(I):OutSwitchStartChB ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):OutSwitchStartChB?  
[:Sweep:Source(1):OutSwitchStartChB ]Value

**Related Command(s):** OutSwitchStartChB

**Description:** Queries the starting logical channel number for the B-channel output switcher sweep.

## OutSwitchStartChB

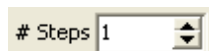
**Command Syntax:** :Sweep:Source(I):OutSwitchStartChB Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Sweep:Source(1):OutSwitchStartChB Value [, AllowCoercion]

**Related Command(s):** OutSwitchStartChB?

**Description:** Sets the starting logical channel number for the B-channel output switcher sweep.



## OutSwitchStep?

**Command Syntax:** :Sweep:Source(I):OutSwitchStep?

**Response Syntax:** [:Sweep:Source(I):OutSwitchStep ]Value

**Response Argument(s):** Value <int>

**Example:** :Sweep:Source(1):OutSwitchStep?  
[:Sweep:Source(1):OutSwitchStep ]Value

**Related Command(s):** OutSwitchStep

**Description:** Queries the logical channel number step size for the output switcher sweep.

## OutSwitchStep

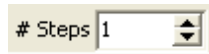
**Command Syntax:** `:Sweep:Source(I):OutSwitchStep Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(1):OutSwitchStep Value [, AllowCoercion]`

**Related Command(s):** OutSwitchStep?

**Description:** Sets the logical channel number step stize for the output switcher sweep.



## SwitchNumSteps?

**Command Syntax:** `:Sweep:Source(I):SwitchNumSteps?`

**Response Syntax:** `[:Sweep:Source(I):SwitchNumSteps ]Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Sweep:Source(1):SwitchNumSteps?`  
`[ :Sweep:Source(1):SwitchNumSteps ]10`

**Related Command(s):** SwitchNumSteps

**Description:** Queries the number of steps in the switcher sweep.

## SwitchNumSteps

**Command Syntax:** `:Sweep:Source(I):SwitchNumSteps Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Sweep:Source(1):SwitchNumSteps 10`

**Related Command(s):** SwitchNumSteps?

**Description:** Sets the number of steps in the switcher sweep.

## 2.3.6.2 Sweep Settling

<b>Object:</b>	<b>:Sweep:Settling(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1 2 3 ... 116 117}
<i>Description:</i>	Commands Related to the settling parameters for the <i>i</i> <sup>th</sup> settler.

The commands below set the properties for the settler :Sweep:Settling(*i*). To find the index *i* corresponding to a given measurement use the command:

```
Instrument:SettlerIndex? MeasId
```

See the [Instrument](#) section for a list of measurement IDs. For instance, to get the settler index corresponding to the FFT spectrum for analyzer A1 send:

```
:Instrument:SettlerIndex? msA1FFTSpectrum
```

the response is:

```
[ :Instrument:SettlerIndex ] 8
```

And the settling parameters for this measurement can then be accessed using:

```
:Sweep:Settling(8):Command
```

	Precision	nPoints	Profile	Threshold	Delay
Level A - Analog	1.0000 %	3	Exponential	100.00 nVrms	30.000 msec

## Delay?

**Command Syntax:** :Sweep:Settling(*I*):Delay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Settling(*I*):Delay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Settling(33):Delay?  
 [:Sweep:Settling(33):Delay] 0.030 s

**Related Command(s):** Delay

**Description:** Queries the delay value for the specified settling object. The delay specifies the amount of time between the start of a new sweep point and when SR1 will begin looking for points that match the settling profile and precision.

## Delay

**Command Syntax:** `:Sweep:Settling(I):Delay Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Settling(41): 0.030 s`

**Related Command(s):** Delay?

**Description:** Sets the delay value for the specified settling object. The delay specifies the amount of time between the start of a new sweep point and when SR1 will begin looking for points that match the settling profile and precision.

## Thresh?

**Command Syntax:** `:Sweep:Settling(I):Thresh? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Sweep:Settling(I):Thresh] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Sweep:Settling(12):Thresh?`  
`[:Sweep:Settling(12):Thresh] 0.010 Vrms`

**Related Command(s):** Thresh

**Description:** Queries the threshold value for the specified settling object. The threshold specifies a minimum precision window to be used for determining settling when the value of the underlying measurement becomes small.

## Thresh

**Command Syntax:** `:Sweep:Settling(I):Thresh Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Sweep:Settling(5):Thresh 0.010 Vrms`

**Related Command(s):** Thresh?

**Description:** Sets the threshold value for the specified settling object. The threshold specifies a minimum precision window to be used for determining settling when the value of the underlying measurement becomes small.

## Method?

**Command Syntax:** `:Sweep:Settling(I):Method?`

**Command Argument(s):**

**Response Syntax:** `[:Sweep:Settling(I):Method] Value`

**Response Argument(s):** `Value <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}`

**Example:** `:Sweep:Settling(89):Method?`  
`[:Sweep:Settling(89):Method] stlNone`

**Related Command(s):** Method

**Description:** Queries the algorithm type for the specified settling object.

## Method

**Command Syntax:** :Sweep:Settling(I):Method *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:Settling(65):Method stlNone

**Related Command(s):** Method?

**Description:** Sets the algorithm type for the specified settling object.

## N?

**Command Syntax:** :Sweep:Settling(I):N?

**Command Argument(s):**

**Response Syntax:** [:Sweep:Settling(I):N] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Sweep:Settling(74):N?  
 [:Sweep:Settling(I):N] 5

**Related Command(s):** N

**Description:** Queries the number of points for the specified settling object.

## N

**Command Syntax:** :Sweep:Settling(I):N *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Sweep:Settling(74):N 5

**Related Command(s):** N?

**Description:** Sets the number of points for the specified settling object.

## Tolerance?

**Command Syntax:** :Sweep:Settling(I):Tolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Sweep:Settling(I):Tolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Sweep:Settling(49):Tolerance?  
 [:Sweep:Settling(49):Tolerance] 1 %

**Related Command(s):** Tolerance

**Description:** Queries the tolerance window for the specified settling object.



## Tolerance

**Command Syntax:** :Sweep:Settling(I):Tolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Sweep:Settling(99):Tolerance 1 pct

**Related Command(s):** Tolerance?

**Description:** Sets the tolerance window for the specified settling object.

### 2.3.7 Analyzer

<b>Object:</b>	<b>:Alyzr(i)</b> <b>:Ana(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the specified analyzer, either A0 or A1.

## Function and Input Commands

### Function?

**Command Syntax:** :Alyzr(I):Function?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(I):Function] Value

**Response Argument(s):** Value <int> {azTimeDomDet=0 | azFFT=1 | azFFT2Ch=2 | azTHD=5 | azIMD=6 | azMultitone=7 | azJitter=9 | azHistogram=10}

**Example:** :Alyzr(0):Function?

[ :Alyzr(0):Function] azTimeDomDet

**Related Command(s):** Function

**Description:** Queries the Function (type) of the specified analyzer.

### Function

**Command Syntax:** :Alyzr(I):Function Value [, AllowCoercion]

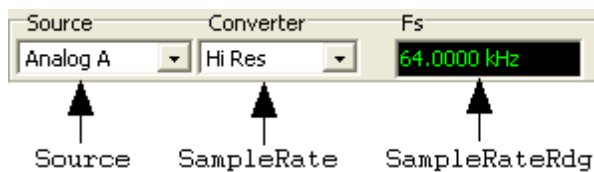
**Command Argument(s):** Value <int> {azTimeDomDet=0 | azFFT=1 | azFFT2Ch=2 | azTHD=5 | azIMD=6 | azMultitone=7 | azJitter=9 | azHistogram=10}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Function azTimeDomDet

**Related Command(s):** Function?

**Description:** Sets the Function (type) of the specified analyzer.



## SampleRate?

**Command Syntax:** :Alyzr(I):SampleRate?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(I):SampleRate] Value

**Response Argument(s):** Value <int> {azHiBW=0 | azHiRes=1 | azISR=2 | azDetOutSR=3}

**Example:** :Alyzr(1):SampleRate?

**[ :Alyzr(1):SampleRate] azHiBW**

**Related Command(s):** Queries the converter/sample rate selection for the given analyzer. The first two selections, HiBw and HiRes, are for analog inputs. ISR is used for Digital Audio inputs. DetOutSR (Detector Output) is used for certain analyzer to set the input to the output of the time domain detector running on the other analyzer.

**Description:** Queries the converter/sample rate selection for the given analyzer.

## SampleRate

**Command Syntax:** :Alyzr(I):SampleRate Value [, AllowCoercion]

**Command Argument(s):** Value <int> {azHiBW=0 | azHiRes=1 | azISR=2 | azDetOutSR=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(1):SampleRate azHiBW

**Related Command(s):** SampleRate?

**Description:** Sets the converter/sample rate selection for the given analyzer. The first two selections, HiBw and HiRes, are for analog inputs. ISR is used for Digital Audio inputs. DetOutSR (Detector Output) is used for certain analyzer to set the input to the output of the time domain detector running on the other analyzer. Note it is not necessary to use the SampleRate command except to choose between the HiResolution and HiBandwidth converters for analog inputs. In all other cases the correct sample rate is automatically chosen when the source for the analyzer is selected.

## SampleRateRdg?

**Command Syntax:** :Alyzr(I):SampleRateRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(I):SampleRateRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(1):SampleRateRdg?

**[ :Alyzr(1):SampleRateRdg] 512000.0 Hz**

**Description:** Queries the effective sample rate for the given analyzer.

## Source?

**Command Syntax:** :Alyzr(I):Source?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(I):Source] Value

**Response Argument(s):** Value <int> {azAnlg=0 | azDig=1 | azAnlgA=2 | azAnlgB=3 | azDigA=4 | azDigB=5 | azDetOut=6}

**Example:** :Alyzr(0):Source?

[ :Alyzr(0):Source ] azAnlg

**Related Command(s):** Source

**Description:** Queries the analyzer input source selection.

## Source

**Command Syntax:** :Alyzr(I):Source Value [, AllowCoercion]

**Command Argument(s):** Value <int> {azAnlg=0 | azDig=1 | azAnlgA=2 | azAnlgB=3 | azDigA=4 | azDigB=5 | azDetOut=6}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Source azAnlg

**Related Command(s):** Source?

**Description:** Sets the analyzer input source selection.



## Analyzer Level Commands

### Analog

#### AnlgFreqARdg?

**Command Syntax:** :Alyzr(I):AnlgFreqARdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgFreqARdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(0):AnlgFreqARdg?

[ :Alyzr(0):AnlgFreqARdg ] 1000.0 Hz

**Description:** Queries the analyzer's analog A-channel frequency reading.

## AnlgFreqBRdg?

**Command Syntax:** :Alyzr(I):AnlgFreqBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgFreqBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr (1) :AnlgFreqBRdg?  
[:Alyzr (1) :AnlgFreqBRdg] 1000.0 Hz

**Description:** Queries the analyzer's analog B-channel frequency reading.

## AnlgLevelARdg?

**Command Syntax:** :Alyzr(I):AnlgLevelARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgLevelARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr (1) :AnlgLevelARdg?  
[:Alyzr (1) :AnlgLevelARdg] 1.2 Vrms

**Description:** Queries the analyzer's analog A-channel level reading..

## AnlgLevelBRdg?

**Command Syntax:** :Alyzr(I):AnlgLevelBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgLevelBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr (0) :AnlgLevelBRdg?  
[:Alyzr (0) :AnlgLevelBRdg] 1.2 Vrms

**Description:** Queries the analyzer's analog B-channel level reading..

## AnlgPhaseRdg?

**Command Syntax:** :Alyzr(I):AnlgPhaseRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgPhaseRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr (0) :AnlgPhaseRdg?  
[:Alyzr (0) :AnlgPhaseRdg] 0.0

**Description:** Queries the analyzer's analog intra-channel phase reading.

## Digital

### DigFreqARdg?

*Command Syntax:* :Alyzr(I):DigFreqARdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:Alyzr(I):DigFreqARdg] *Value*

*Response Argument(s):* *Value* <unit>

*Example:* :Alyzr(0):DigFreqARdg?  
[:Alyzr(0):DigFreqARdg] 997.0 Hz

*Description:* Queries the analyzer's digital A-channel frequency reading.

### DigFreqBRdg?

*Command Syntax:* :Alyzr(I):DigFreqBRdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:Alyzr(I):DigFreqBRdg] *Value*

*Response Argument(s):* *Value* <unit>

*Example:* :Alyzr(1):DigFreqBRdg?  
[:Alyzr(1):DigFreqBRdg] 997.0 Hz

*Description:* Queries the analyzer's digital B-channel frequency reading.

### DigLevelARdg?

*Command Syntax:* :Alyzr(I):DigLevelARdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:Alyzr(I):DigLevelARdg] *Value*

*Response Argument(s):* *Value* <unit>

*Example:* :Alyzr(1):DigLevelARdg?  
[:Alyzr(1):DigLevelARdg] 0.8 FFS

*Description:* Queries the analyzer's digital A-channel level reading..

### DigLevelBRdg?

*Command Syntax:* :Alyzr(I):DigLevelBRdg? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:Alyzr(I):DigLevelBRdg] *Value*

*Response Argument(s):* *Value* <unit>

*Example:* :Alyzr(0):DigLevelBRdg?  
[:Alyzr(0):DigLevelBRdg] Value

*Description:* Queries the analyzer's digital B-channel level reading..

## DigPhaseRdg?

**Command Syntax:** :Alyzr(I):DigPhaseRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):DigPhaseRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):DigPhaseRdg?

[ :Alyzr(0):DigPhaseRdg] **Value**

**Description:** Queries the analyzer's digital audio intra-channel phase reading.

## Analyzer Reference Commands

### Analog



Note that even though the reference commands belong are presented as belonging to the individual analyzers A0 and A1 there are in reality only one set of references for both analyzers. Thus :Alyzer(0):AnlgdBrA and :Alyzer(1):AnlgdBrA both change the same quantity.

## AnlgdBrA?

**Command Syntax:** :Alyzr(I):AnlgdBrA? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgdBrA] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(1):AnlgdBrA?

[ :Alyzr(1):AnlgdBrA] **1.000 Vrms**

**Related Command(s):** AnlgdBrA

**Description:** Queries the dBrA reference for the both analyzers.

## AnlgdBrA

**Command Syntax:** :Alyzr(I):AnlgdBrA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(1):AnlgdBrA 1.000 Vrms

**Related Command(s):** AnlgdBrA?

**Description:** Sets the dBrA reference for the both analyzers.

## AnlgdBrB?

**Command Syntax:** :Alyzr(I):AnlgdBrB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgdBrB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(1):AnlgdBrB?  
[:Alyzr(1):AnlgdBrB]1.000 Vrms

**Related Command(s):** AnlgdBrB

**Description:** Queries the dBrB reference for the both analyzers.

## AnlgdBrB

**Command Syntax:** :Alyzr(I):AnlgdBrB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(1):AnlgdBrB 1.000 Vrms

**Related Command(s):** AnlgdBrB?

**Description:** Sets the dBrB reference for both analyzers.

## AnlgFreq?

**Command Syntax:** :Alyzr(I):AnlgFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):AnlgFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):AnlgFreq?  
[:Alyzr(0):AnlgFreq] 1000.0 Hz

**Related Command(s):** AnlgFreq

**Description:** Queries the analog frequency reference for both analyzers.

## AnlgFreq

**Command Syntax:** :Alyzr(I):AnlgFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):AnlgFreq 1000.0 Hz

**Related Command(s):** AnlgFreq?

**Description:** Sets the analog frequency reference for both analyzers.



## dBmZ?

**Command Syntax:** :Alyzr(I):dBmZ? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):dBmZ] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):dBmZ?

**[ :Alyzr(0):dBmZ ] 600 ohms**

**Related Command(s):** dBmZ

**Description:** Queries the dBm reference impedance for both analyzers.

## dBmZ

**Command Syntax:** :Alyzr(I):dBmZ *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):dBmZ 600 ohms

**Related Command(s):** dBmZ?

**Description:** Sets the dBm reference impedance for both analyzers.

## WattsZ?

**Command Syntax:** :Alyzr(I):WattsZ? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):WattsZ] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(1):WattsZ?

**[ :Alyzr(1):WattsZ ] Value**

**Related Command(s):** WattsZ

**Description:** Queries the Watts reference impedance for both analyzers.

## WattsZ

**Command Syntax:** :Alyzr(I):WattsZ *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(1):WattsZ *Value*

**Related Command(s):** WattsZ?

**Description:** Sets the Watts reference impedance for both analyzers.

## Digital Audio

Digital

V/Fs 1.0000 Vrms

dBrA 1.0000 FFS

dBrB 1.0000 FFS

Freq 1.0000 kHz

## DigdBrA?

**Command Syntax:** :Alyzr(I):DigdBrA? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):DigdBrA] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(1):DigdBrA?

[ :Alyzr(1):DigdBrA] 1.000 Vrms

**Related Command(s):** DigdBrA

**Description:** Queries the digital audio dBrA reference value for both analyzers.

## DigdBrA

**Command Syntax:** :Alyzr(I):DigdBrA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(1):DigdBrA 1.000 Vrms

**Related Command(s):** DigdBrA?

**Description:** Sets the digital audio dBrA reference value for both analyzers.

## DigdBrB?

**Command Syntax:** :Alyzr(I):DigdBrB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(I):DigdBrB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):DigdBrB?

[ :Alyzr(0):DigdBrB] 1.000 Vrms

**Related Command(s):** DigdBrB

**Description:** Queries the digital audio dBrB reference value for both analyzers.

## DigdBrB

**Command Syntax:** :Alyzr(I):DigdBdB Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):DigdBdB 1.0 Vrms

**Related Command(s):** DigdBdB?

**Description:** Sets the digital audio dBdB reference value for both analyzers.

## DigFreq?

**Command Syntax:** :Alyzr(I):DigFreq? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(I):DigFreq] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(0):DigFreq?  
[:Alyzr(0):DigFreq] 100.0 Hz

**Related Command(s):** DigFreq

**Description:** Queries the digital audio frequency reference for both analyzers.

## DigFreq

**Command Syntax:** :Alyzr(I):DigFreq Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):DigFreq 100 Hz

**Related Command(s):** DigFreq?

**Description:** Sets the digital audio frequency reference for both analyzers.

## DigVfs?

**Command Syntax:** :Alyzr(I):DigVfs? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(I):DigVfs] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(1):DigVfs?  
[:Alyzr(1):DigVfs] 10 Vrms

**Related Command(s):** DigVfs

**Description:** Queries the digital audio Volts/Full Scale value for both analyzers.

## DigVfs

*Command Syntax:* :Alyzr(I):DigVfs Value [, AllowCoercion]

*Command Argument(s):* Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

*Example:* :Alyzr(1):DigVfs 10 Vrms

*Related Command(s):* DigVfs?

*Description:* Sets the digital audio Volts/Full Scale value for both analyzers.

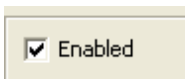
## Supported Form Commands:

The form commands for :Alyzr(i) refer to the currently active sub-analyzer.

**:Alyzr(i):OpenForm**  
**:Alyzr(i):OpenFormwID?**  
**:Alyzr(i):CloseForm**  
**:Alyzr(i):CloseForms**  
**:Alyzr(i):FormCount?**  
**:Alyzr(i):FormID?**

2.3.7.1 Analyzer Trigger

<b>Object:</b>	<b>:Alyzr(i):Trigger</b> <b>:Alyzr(i):Trg</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the triggering parameters for the specified analyzer.



**Active?**

*Command Syntax:* :Alyzr(i):Trigger:Active?

*Command Argument(s):*

*Response Syntax:* [:Alyzr(i):Trigger:Active] Value

*Response Argument(s):* Value <int> {False=0|True=1}

*Example:* :Alyzr(0):Trigger:Active?  
[:Alyzr(0):Trigger:Active] False

*Related Command(s):* Active

*Description:* Queries the trigger enabled state for the given analyzer.

**Active**

*Command Syntax:* :Alyzr(i):Trigger:Active Value [, AllowCoercion]

*Command Argument(s):* Value <int> {False=0|True=1}  
AllowCoercion <bool> {False=0|True=1}

*Example:* :Alyzr(0):Trigger:Active False

*Related Command(s):* Active?

*Description:* Sets the trigger enabled state for the given analyzer.



**Source?**

*Command Syntax:* :Alyzr(i):Trigger:Source?

*Command Argument(s):*

*Response Syntax:* [:Alyzr(i):Trigger:Source] Value

*Response Argument(s):* Value <int> {trgInputCh=0|trgOtherCh=1|trgExt=2|trgManual=3|trgGen=4|trgDigAudBlock=5|trgBurstA=6|trgBurstB=7}

*Example:* :Alyzr(0):Trigger:Source?  
[:Alyzr(0):Trigger:Source] trgInputCh

*Related Command(s):* Source

*Description:* Queries the trigger source for the given analyzer.

## Source

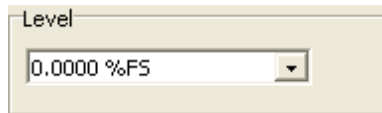
**Command Syntax:** `:Alyzr(i):Trigger:Source Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {trgInputCh=0 | trgOtherCh=1 | trgExt=2 | trgManual=3 | trgGen=4 | trgDigAudBlock=5 | trgBurstA=6 | trgBurstB=7}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):Trigger:Source trgInputCh`

**Related Command(s):** Source?

**Description:** Sets the trigger source for the given analyzer.



## AnlgLevelA?

**Command Syntax:** `:Alyzr(i):Trigger:AnlgLevelA? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Trigger:AnlgLevelA] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Trigger:AnlgLevelA?`  
`[:Alyzr(0):Trigger:AnlgLevelA] 50.0 pctFS`

**Related Command(s):** AnlgLevelA

**Description:** Queries the A-channel analog trigger level.

## AnlgLevelA

**Command Syntax:** `:Alyzr(i):Trigger:AnlgLevelA Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):Trigger:AnlgLevelA 50.0 pctFS`

**Related Command(s):** AnlgLevelA?

**Description:** Sets the A-channel analog trigger level.

## AnlgLevelB?

**Command Syntax:** `:Alyzr(i):Trigger:AnlgLevelB? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Trigger:AnlgLevelB] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Trigger:AnlgLevelB?`  
`[:Alyzr(0):Trigger:AnlgLevelB] 50.0 pctFS`

**Related Command(s):** AnlgLevelB

**Description:** Queries the B-channel analog trigger level.

## AnlgLevelB

**Command Syntax:** :Alyzr(i):Trigger:AnlgLevelB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Trigger:AnlgLevelB 50.0 pctFS

**Related Command(s):** AnlgLevelB?

**Description:** Sets the B-channel analog trigger level.

## DigLevelA?

**Command Syntax:** :Alyzr(i):Trigger:DigLevelA? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Trigger:DigLevelA] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Trigger:DigLevelA?  
[:Alyzr(0):Trigger:DigLevelA] 40 pctFS

**Related Command(s):** DigLevelA

**Description:** Queries the A-Channel digital audio trigger level.

## DigLevelA

**Command Syntax:** :Alyzr(i):Trigger:DigLevelA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Trigger:DigLevelA 40 pctFS

**Related Command(s):** DigLevelA?

**Description:** Sets the A-Channel digital audio trigger level.

## DigLevelB?

**Command Syntax:** :Alyzr(i):Trigger:DigLevelB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Trigger:DigLevelB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Trigger:DigLevelB?  
[:Alyzr(0):Trigger:DigLevelB] 40 pctFS

**Related Command(s):** DigLevelB

**Description:** Queries the B-Channel digital audio trigger level.

## DigLevelB

**Command Syntax:** :Alyzr(i):Trigger:DigLevelB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Trigger:DigLevelB 40 pctFS

**Related Command(s):** DigLevelB?

**Description:** Sets the B-Channel digital audio trigger level.

## JitterLevel?

**Command Syntax:** :Alyzr(i):Trigger:JitterLevel? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Trigger:JitterLevel] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Trigger:JitterLevel?  
 [:Alyzr(0):Trigger:JitterLevel] 50 pctFS

**Related Command(s):** JitterLevel

**Description:** Queries the trigger level when the analyzer is set to Jitter Analyzer.

## JitterLevel

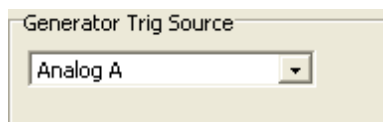
**Command Syntax:** :Alyzr(i):Trigger:JitterLevel *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Trigger:JitterLevel 50 pctFS

**Related Command(s):** JitterLevel?

**Description:** Sets the trigger level when the analyzer is set to Jitter Analyzer.



## GenTriggerSource?

**Command Syntax:** :Alyzr(i):Trigger:GenTriggerSource?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Trigger:GenTriggerSource] *Value*

**Response Argument(s):** *Value* <int> {trgAnlgA=0|trgAnlgB=1|trgDigA=2|trgDigB=3|trgJitter=4}

**Example:** :Alyzr(0):Trigger:GenTriggerSource?  
 [:Alyzr(0):Trigger:GenTriggerSource] trgAnlgA

**Related Command(s):** GenTriggerSource

**Description:** Queries the generator trigger source.



## GenTriggerSource

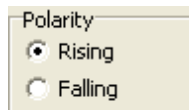
**Command Syntax:** `:Alyzr(i):Trigger:GenTriggerSource Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {trgAnlgA=0 | trgAnlgB=1 | trgDigA=2 | trgDigB=3 | trgJitter=4}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Trigger:GenTriggerSource trgAnlgA`

**Related Command(s):** GenTriggerSource?

**Description:** Sets the generator trigger source.



## Polarity?

**Command Syntax:** `:Alyzr(i):Trigger:Polarity?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Trigger:Polarity] Value`

**Response Argument(s):** `Value <int> {trgRising=0 | trgFalling=1}`

**Example:** `:Alyzr(0):Trigger:Polarity?`

`[:Alyzr(0):Trigger:Polarity] trgRising`

**Related Command(s):** Polarity

**Description:** Queries the trigger polarity.

## Polarity

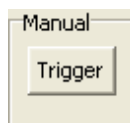
**Command Syntax:** `:Alyzr(i):Trigger:Polarity Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {trgRising=0 | trgFalling=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Trigger:Polarity trgRising`

**Related Command(s):** Polarity?

**Description:** Sets the trigger polarity.



## ManTrig

**Command Syntax:** `:Alyzr(i):Trigger:ManTrig`

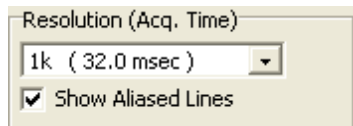
**Command Argument(s):** None

**Example:** `:Alyzr(0):Trigger:ManTrig`

**Description:** Performs a manual trigger.

## 2.3.7.2 FFT1 Analyzer

<b>Object:</b>	<b>:Alyzr(i):FFT</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the Single Channel FFT (FFT1) analyzer belonging to either A0 or A1.

**Lines?**

**Command Syntax:** :Alyzr(i):FFT:Lines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT:Lines] Value

**Response Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

**Example:** :Alyzr(0):FFT:Lines?  
[:Alyzr(0):FFT:Lines] fft132k

**Related Command(s):** Lines

**Description:** Queries the number of FFT lines (resolution).

**Lines**

**Command Syntax:** :Alyzr(i):FFT:Lines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:Lines fft132k

**Related Command(s):** Lines?

**Description:** Sets the number of FFT lines (resolution).

**ShowAllLines?**

**Command Syntax:** :Alyzr(i):FFT>ShowAllLines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT>ShowAllLines] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Alyzr(0):FFT>ShowAllLines?  
[:Alyzr(0):FFT>ShowAllLines] False

**Related Command(s):** ShowAllLines

**Description:** Queries the state of the "Show Aliased Lines" checkbox.

## ShowAllLines

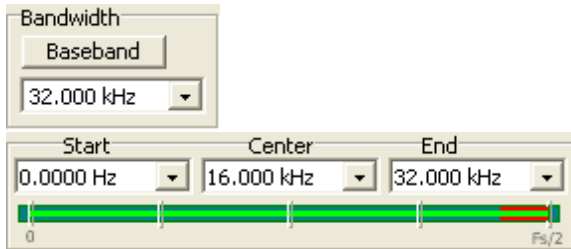
**Command Syntax:** `:Alyzr(i):FFT:ShowAllLines Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):FFT:ShowAllLines False`

**Related Command(s):** ShowAllLines?

**Description:** Sets the state of the "Show Aliased Lines" checkbox.



## Span?

**Command Syntax:** `:Alyzr(i):FFT:Span?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):FFT:Span] Value`

**Response Argument(s):** `Value <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}`

**Example:** `:Alyzr(0):FFT:Span?`

`[:Alyzr(0):FFT:Span] fftFsDiv2`

**Related Command(s):** Span

**Description:** Queries the frequency span (bandwidth) setting of the FFT analyzer. The maximum span is  $F_s/2$ , represented by the enumeration `fftFsDiv2`. Each subsequent setting divides the frequency span by 2.

## Span

**Command Syntax:** `:Alyzr(i):FFT:Span Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):FFT:Span fftFsDiv2`

**Related Command(s):** Span?

**Description:** Sets the frequency span (bandwidth) setting of the FFT analyzer. The maximum span is  $F_s/2$ , represented by the enumeration `fftFsDiv2`. Each subsequent setting divides the frequency span by 2..

## Baseband

**Command Syntax:** :Alyzr(i):FFT:Baseband

**Command Argument(s):** None

**Example:** :Alyzr(0):FFT:Baseband

**Description:** Sets the FFT analyzer to its maximum frequency range.

## StartFreq?

**Command Syntax:** :Alyzr(i):FFT:StartFreq? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(i):FFT:StartFreq] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(0):FFT:StartFreq?  
[:Alyzr(0):FFT:StartFreq] 0 Hz

**Related Command(s):** StartFreq

**Description:** Queries the value of the lowest frequency in the analysis region.

## StartFreq

**Command Syntax:** :Alyzr(i):FFT:StartFreq Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT:StartFreq 0 Hz

**Related Command(s):** StartFreq?

**Description:** Sets the value of the lowest frequency in the analysis region.

## CenterFreq?

**Command Syntax:** :Alyzr(i):FFT:CenterFreq? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(i):FFT:CenterFreq] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(0):FFT:CenterFreq?  
[:Alyzr(0):FFT:CenterFreq] 16000.0 Hz

**Related Command(s):** CenterFreq

**Description:** Queries the value of the midpoint of the analysis region.

## CenterFreq

**Command Syntax:** :Alyzr(i):FFT:CenterFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:CenterFreq *Value*

**Related Command(s):** CenterFreq?

**Description:** Sets the value of the midpoint of the analysis region.

## StopFreq?

**Command Syntax:** :Alyzr(i):FFT:StopFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT:StopFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT:StopFreq?  
[:Alyzr(0):FFT:StopFreq] 32000 Hz

**Related Command(s):** StopFreq

**Description:** Queries the value of the highest frequency in the analysis region.

## StopFreq

**Command Syntax:** :Alyzr(i):FFT:StopFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:StopFreq 32000 Hz

**Related Command(s):** StopFreq?

**Description:** Sets the value of the highest frequency in the analysis region.

## TimeRecordDuration?

**Command Syntax:** :Alyzr(i):FFT:TimeRecordDuration? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

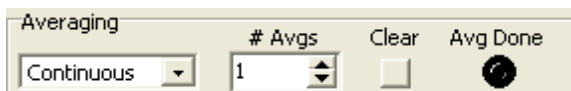
**Response Syntax:** [:Alyzr(i):FFT:TimeRecordDuration] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT:TimeRecordDuration?  
[:Alyzr(0):FFT:TimeRecordDuration] 0.004 s

**Related Command(s):**

**Description:** Queries the length of the time record for the given span and resolution settings.



## Averaging?

**Command Syntax:** `:Alyzr(i):FFT:Averaging?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):FFT:Averaging] Value`

**Response Argument(s):** `Value <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2}`

**Example:** `:Alyzr(0):FFT:Averaging?`

`[ :Alyzr(0):FFT:Averaging] avgNone`

**Related Command(s):** Averaging

**Description:** Queries the averaging mode for the FFT analyzer.

## Averaging

**Command Syntax:** `:Alyzr(i):FFT:Averaging Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):FFT:Averaging avgNone`

**Related Command(s):** Averaging?

**Description:** Sets the averaging mode for the FFT analyzer.

## AvgDone?

**Command Syntax:** `:Alyzr(i):FFT:AvgDone?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):FFT:AvgDone] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):FFT:AvgDone?`

`[ :Alyzr(0):FFT:AvgDone] False`

**Description:** Queries the "Average Done" status of the FFT 1analyzer.

## NumAverages?

**Command Syntax:** `:Alyzr(i):FFT:NumAverages?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):FFT:NumAverages] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Alyzr(0):FFT:NumAverages?`

`[ :Alyzr(0):FFT:NumAverages] Value`

**Related Command(s):** NumAverages

**Description:** Queries the number of averages used by the analyzer.

## NumAverages

**Command Syntax:** :Alyzr(i):FFT:NumAverages *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:NumAverages *Value*

**Related Command(s):** NumAverages?

**Description:** Sets the number of averages used by the analyzer.

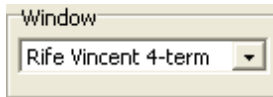
## ResetAvg

**Command Syntax:** :Alyzr(i):FFT:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):FFT:ResetAvg

**Description:** Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



## Window?

**Command Syntax:** :Alyzr(i):FFT:Window?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT:Window] *Value*

**Response Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
 fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
 fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}

**Example:** :Alyzr(0):FFT:Window?

**[ :Alyzr(0) :FFT:Window] fftBlackmanHarris**

**Related Command(s):** Window

**Description:** Queries the window selection for the analyzer.

## Window

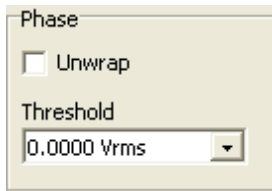
**Command Syntax:** :Alyzr(i):FFT:Window *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
 fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
 fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:Window fftBlackmanHarris

**Related Command(s):** Window?

**Description:** Sets the window selection for the analyzer.



## AnlgPhaseThreshold?

**Command Syntax:** :Alyzr(i):FFT:AnlgPhaseThreshold? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT:AnlgPhaseThreshold] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT:AnlgPhaseThreshold?

[ :Alyzr(0):FFT:AnlgPhaseThreshold] 0.0001 Vrms

**Related Command(s):** AnlgPhaseThreshold

**Description:** Queries the amplitude threshold for calculating phase for analog inputs.

## AnlgPhaseThreshold

**Command Syntax:** :Alyzr(i):FFT:AnlgPhaseThreshold *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT:AnlgPhaseThreshold 0.0001 Vrms

**Related Command(s):** AnlgPhaseThreshold?

**Description:** Set the amplitude threshold for calculating phase for analog inputs.

## DigPhaseThreshold?

**Command Syntax:** :Alyzr(i):FFT:DigPhaseThreshold? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT:DigPhaseThreshold] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT:DigPhaseThreshold?

[ :Alyzr(0):FFT:DigPhaseThreshold] *Value*

**Related Command(s):** DigPhaseThreshold

**Description:** Queries the amplitude threshold for calculating phase for digital audio inputs.



## DigPhaseThreshold

**Command Syntax:** :Alyzr(i):FFT:DigPhaseThreshold *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:DigPhaseThreshold *Value*

**Related Command(s):** DigPhaseThreshold?

**Description:** Sets the amplitude threshold for calculating phase for digital audio inputs.

## PhaseUnwrap?

**Command Syntax:** :Alyzr(i):FFT:PhaseUnwrap?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT:PhaseUnwrap] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:PhaseUnwrap?  
 [:Alyzr(0):FFT:PhaseUnwrap] **False**

**Related Command(s):** PhaseUnwrap

**Description:** Queries the Phase Unwrap status for the analyzer.

## PhaseUnwrap

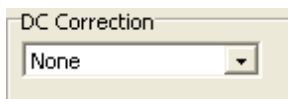
**Command Syntax:** :Alyzr(i):FFT:PhaseUnwrap *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:PhaseUnwrap **False**

**Related Command(s):** PhaseUnwrap?

**Description:** Sets Phase Unwrap on and off for the analyzer.



## DCCorrectionMode?

**Command Syntax:** :Alyzr(i):FFT:DCCorrectionMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT:DCCorrectionMode] *Value*

**Response Argument(s):** *Value* <int> {dcmNone=0 | dcmMean=1 | dcmPkPk=2}

**Example:** :Alyzr(0):FFT:DCCorrectionMode?  
 [:Alyzr(0):FFT:DCCorrectionMode] **dcmNone**

**Related Command(s):** DCCorrectionMode

**Description:** Queries the DC Correction mode setting for the analyzer.

## DCCorrectionMode

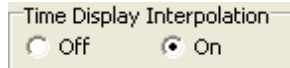
**Command Syntax:** :Alyzr(i):FFT:DCCorrectionMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {dcmNone=0 | dcmMean=1 | dcmPkPk=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:DCCorrectionMode dcmNone

**Related Command(s):** DCCorrectionMode?

**Description:** Sets the DC Correction mode setting for the analyzer.



## DisplayInterpolation?

**Command Syntax:** :Alyzr(i):FFT:DisplayInterpolation?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT:DisplayInterpolation] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:DisplayInterpolation?

**[:Alyzr(0):FFT:DisplayInterpolation] False**

**Related Command(s):** DisplayInterpolation

**Description:** Queries the time-domain display interpolation status for the analyzer.

## DisplayInterpolation

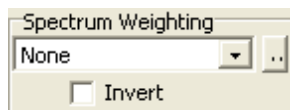
**Command Syntax:** :Alyzr(i):FFT:DisplayInterpolation *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT:DisplayInterpolation False

**Related Command(s):** DisplayInterpolation?

**Description:** Sets the time-domain display interpolation status for the analyzer.



## SetEqFile

**Command Syntax:** :Alyzr(i):FFT:SetEqFile *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Alyzr(0):FFT:SetEqFile AWeighting.EQ

**Description:** Sets an EQ file to weight the FFT spectrum. The argument is the file name, including the suffix. The default directory is "user/eqCurves". To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e.

:Alyzr(i):FFT:SetEqFile ""

## InvertEq?

**Command Syntax:** `:Alyzr(i):FFT:InvertEq?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):FFT:InvertEq] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr (0) :FFT:InvertEq?  
[:Alyzr (0) :FFT:InvertEq] False`

**Related Command(s):** InvertEq

**Description:** Queries the invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.

## InvertEq

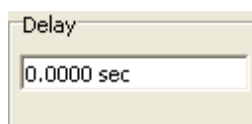
**Command Syntax:** `:Alyzr(i):FFT:InvertEq Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr (0) :FFT:InvertEq False`

**Related Command(s):** InvertEq?

**Description:** Sets the invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.



## TriggerDelay?

**Command Syntax:** `:Alyzr(i):FFT:TriggerDelay? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):FFT:TriggerDelay] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr (0) :FFT:TriggerDelay?  
[:Alyzr (0) :FFT:TriggerDelay] 0.0000325 s`

**Related Command(s):** TriggerDelay

**Description:** Queries the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample, (1/Fs). Negative (pre-trigger) delays are allowed.

## TriggerDelay

**Command Syntax:** :Alyzr(i):FFT:TriggerDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT:TriggerDelay 0.0000325 s

**Related Command(s):** TriggerDelay?

**Description:** Sets the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample, (1/Fs). Negative (pre-trigger) delays are allowed.

## Form Commands:

**:Alyzr(i):FFT:OpenForm**

**:Alyzr(i):FFT:OpenFormwID?**

**:Alyzr(i):FFT:CloseForm**

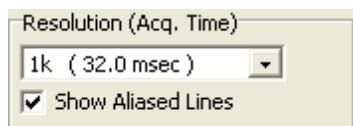
**:Alyzr(i):FFT:CloseForms**

**:Alyzr(i):FFT:FormCount?**

**:Alyzr(i):FFT:FormID?**

### 2.3.7.3 FFT2 Analyzer

<b>Object:</b>	<b>:Alyzr(i):FFT2</b> <b>:Alyzr(i):FFT2Ch</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the Dual Channel FFT analyzer (FFT2) belonging to either A0 or A1.



## Lines?

**Command Syntax:** :Alyzr(i):FFT2Ch:Lines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:Lines] Value

**Response Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

**Example:** :Alyzr(0):FFT2Ch:Lines?  
[:Alyzr(0):FFT2Ch:Lines] fft132k

**Related Command(s):** Lines

**Description:** Queries the number of FFT lines (resolution).

## Lines

**Command Syntax:** :Alyzr(i):FFT2Ch:Lines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:Lines fft132k

**Related Command(s):** Lines?

**Description:** Sets the number of FFT lines (resolution).

## ShowAllLines?

**Command Syntax:** :Alyzr(i):FFT2Ch:ShowAllLines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:ShowAllLines] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:ShowAllLines?

**[ :Alyzr (0) :FFT2Ch:ShowAllLines] False**

**Related Command(s):** ShowAllLines

**Description:** Queries the state of the "Show Aliased Lines" checkbox.

## ShowAllLines

**Command Syntax:** :Alyzr(i):FFT2Ch:ShowAllLines *Value* [, *AllowCoercion*]

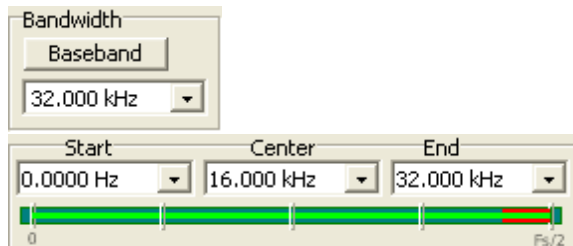
**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:ShowAllLines False

**Related Command(s):** ShowAllLines?

**Description:** Sets the state of the "Show Aliased Lines" checkbox.



## Span?

**Command Syntax:** :Alyzr(i):FFT2Ch:Span?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:Span] *Value*

**Response Argument(s):** *Value* <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}

**Example:** :Alyzr (0) :FFT2Ch:Span?

**[ :Alyzr (0) :FFT2Ch:Span] fftFsDiv2**

**Related Command(s):** Span

**Description:** Queries the frequency span( bandwidth) settling of the FFT2 analyzer. The maximum span is  $F_s/2$ , represented by the enumeration `fftFsDiv2`. Each subsequent setting divides the frequency span by 2.

## Span

**Command Syntax:** `:Alyzr(i):FFT2Ch:Span Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):FFT2Ch:Span fftFsDiv2`

**Related Command(s):** Span?

**Description:** Sets the frequency span( bandwidth) settling of the FFT2 analyzer. The maximum span is  $F_s/2$ , represented by the enumeration `fftFsDiv2`. Each subsequent setting divides the frequency span by 2.

## Baseband

**Command Syntax:** `:Alyzr(i):FFT2Ch:Baseband`

**Command Argument(s):** None

**Example:** `:Alyzr(0):FFT2Ch:Baseband`

**Description:** Sets the FFT2 analyzer frequency span to its maximum value.

## StartFreq?

**Command Syntax:** `:Alyzr(i):FFT2Ch:StartFreq? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):FFT2Ch:StartFreq] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):FFT2Ch:StartFreq?  
[:Alyzr(0):FFT2Ch:StartFreq] 0 Hz`

**Related Command(s):** StartFreq

**Description:** Queries the value of the lowest frequency in the analysis range.

## StartFreq

**Command Syntax:** `:Alyzr(i):FFT2Ch:StartFreq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):FFT2Ch:StartFreq 0 Hz`

**Related Command(s):** StartFreq?

**Description:** Sets the value of the lowest frequency in the analysis range.

## CenterFreq?

**Command Syntax:** :Alyzr(i):FFT2Ch:CenterFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:CenterFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:CenterFreq?  
[:Alyzr(0):FFT2Ch:CenterFreq] 50000 Hz

**Related Command(s):** CenterFreq

**Description:** Queries the value of the midpoint of the analysis range.

## CenterFreq

**Command Syntax:** :Alyzr(i):FFT2Ch:CenterFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:CenterFreq 50000 Hz

**Related Command(s):** CenterFreq?

**Description:** Sets the value of the midpoint of the analysis range.

## StopFreq?

**Command Syntax:** :Alyzr(i):FFT2Ch:StopFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:StopFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:StopFreq?  
[:Alyzr(0):FFT2Ch:StopFreq] 200000 Hz

**Related Command(s):** StopFreq

**Description:** Queries the value of the highest frequency in the analysis range.

## StopFreq

**Command Syntax:** :Alyzr(i):FFT2Ch:StopFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:StopFreq 200000 Hz

**Related Command(s):** StopFreq?

**Description:** Sets the value of the highest frequency in the analysis range.



## TimeRecordDuration?

**Command Syntax:** :Alyzr(i):FFT2Ch:TimeRecordDuration? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

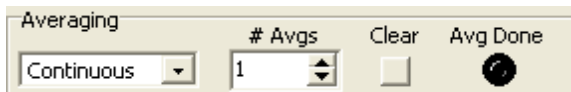
**Response Syntax:** [:Alyzr(i):FFT2Ch:TimeRecordDuration] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:TimeRecordDuration?  
[:Alyzr(0):FFT2Ch:TimeRecordDuration] 0.004 s

**Related Command(s):** TimeRecordDuration

**Description:** Queries the length of the time record for the current span and resolution settings.



## Averaging?

**Command Syntax:** :Alyzr(i):FFT2Ch:Averaging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:Averaging] *Value*

**Response Argument(s):** *Value* <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2}

**Example:** :Alyzr(0):FFT2Ch:Averaging?  
[:Alyzr(0):FFT2Ch:Averaging] avgNone

**Related Command(s):** Averaging

**Description:** Queries the averaging mode for the FFT2 analyzer.

## Averaging

**Command Syntax:** :Alyzr(i):FFT2Ch:Averaging *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:Averaging avgNone

**Related Command(s):** Averaging?

**Description:** Sets the averaging mode for the FFT2 analyzer.

## AvgDone?

**Command Syntax:** :Alyzr(i):FFT2Ch:AvgDone?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:AvgDone] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:AvgDone?  
[:Alyzr(0):FFT2Ch:AvgDone] False

**Description:** Queries the "Average Done" status of the FFT2 analyzer.

## NumAverages?

**Command Syntax:** :Alyzr(i):FFT2Ch:NumAverages?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:NumAverages] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):FFT2Ch:NumAverages?  
[:Alyzr(0):FFT2Ch:NumAverages] 10

**Related Command(s):** NumAverages

**Description:** Queries the number of averages used by the FFT2 analyzer.

## NumAverages

**Command Syntax:** :Alyzr(i):FFT2Ch:NumAverages *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:NumAverages 10

**Related Command(s):** NumAverages?

**Description:** Sets the number of averages used by the FFT2 analyzer.

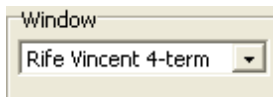
## ResetAvg

**Command Syntax:** :Alyzr(i):FFT2Ch:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):FFT2Ch:ResetAvg

**Description:** Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



## Window?

**Command Syntax:** :Alyzr(i):FFT2Ch:Window?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:Window] *Value*

**Response Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
fftFlatTop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}

**Example:** :Alyzr(0):FFT2Ch:Window?  
[:Alyzr(0):FFT2Ch:Window] fftBlackmanHarris

**Related Command(s):** Window

**Description:** Queries the window selection for the analyzer.

## Window

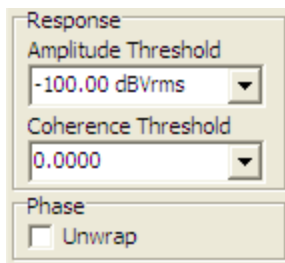
**Command Syntax:** `:Alyzr(i):FFT2Ch:Window Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 | fftFlatTop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 | fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}`  
**AllowCoercion** `<bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):FFT2Ch:Window fftBlackmanHarris`

**Related Command(s):** Window?

**Description:** Sets the window selection for the analyzer.



## AnlgRespThreshold?

**Command Syntax:** `:Alyzr(i):FFT2Ch:AnlgRespThreshold? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):FFT2Ch:AnlgRespThreshold] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):FFT2Ch:AnlgRespThreshold?`  
`[ :Alyzr(0):FFT2Ch:AnlgRespThreshold] 0.0001`

**Related Command(s):** AnlgRespThreshold

**Description:** Queries the Analog magnitude threshold for frequency response computation.

## AnlgRespThreshold

**Command Syntax:** `:Alyzr(i):FFT2Ch:AnlgRespThreshold Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
**AllowCoercion** `<bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):FFT2Ch:AnlgRespThreshold 0.0001`

**Related Command(s):** AnlgRespThreshold?

**Description:** Sets the Analog magnitude threshold for frequency response computation. The analyzer will report 0 magnitude and phase for any frequency for which the A- or B-channel FFT magnitude is below this threshold.

## DigRespThreshold?

**Command Syntax:** :Alyzr(i):FFT2Ch:DigRespThreshold? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:DigRespThreshold] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:DigRespThreshold? dBFS  
[:Alyzr(0):FFT2Ch:DigRespThreshold] -120

**Related Command(s):** DigRespThreshold

**Description:** Queries the Digital magnitude threshold for frequency response computation.

## DigRespThreshold

**Command Syntax:** :Alyzr(i):FFT2Ch:DigRespThreshold *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:DigRespThreshold -120 dBFS

**Related Command(s):** DigRespThreshold?

**Description:** Sets the Digital magnitude threshold for frequency response computation. The analyzer will report 0 magnitude and phase for any frequency for which the A- or B-channel FFT magnitude is below this threshold.

## CoherenceRespThreshold?

**Command Syntax:** :Alyzr(i):FFT2Ch:CoherenceRespThreshold? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:CoherenceRespThreshold] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:CoherenceRespThreshold?  
[:Alyzr(0):FFT2Ch:CoherenceRespThreshold] 0.5

**Related Command(s):** CoherenceRespThreshold

**Description:** Queries the Coherence threshold for frequency response computation.

## CoherenceRespThreshold

**Command Syntax:** :Alyzr(i):FFT2Ch:CoherenceRespThreshold? *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:CoherenceRespThreshold 0.5

**Related Command(s):** CoherenceRespThreshold?

**Description:** Sets the Coherence threshold for frequency response computation. The analyzer will report 0 magnitude and phase for any frequency at which the computed 2-channel coherence is below this threshold.

## PhaseUnwrap?

**Command Syntax:** :Alyzr(i):FFT2Ch:PhaseUnwrap?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:PhaseUnwrap] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:PhaseUnwrap?

**[ :Alyzr (0) :FFT2Ch:PhaseUnwrap] False**

**Related Command(s):** PhaseUnwrap

**Description:** Queries the phase unwrap setting for the FFT2 analyzer.

## PhaseUnwrap

**Command Syntax:** :Alyzr(i):FFT2Ch:PhaseUnwrap *Value* [, *AllowCoercion*]

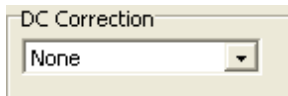
**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:PhaseUnwrap False

**Related Command(s):** PhaseUnwrap?

**Description:** Sets the phase unwrap setting for the FFT2 analyzer.



## DCCorrectionMode?

**Command Syntax:** :Alyzr(i):FFT2Ch:DCCorrectionMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:DCCorrectionMode] *Value*

**Response Argument(s):** *Value* <int> {dcmNone=0 | dcmMean=1 | dcmPkPk=2}

**Example:** :Alyzr (0) :FFT2Ch:DCCorrectionMode?

**[ :Alyzr (0) :FFT2Ch:DCCorrectionMode] dcmNone**

**Related Command(s):** DCCorrectionMode

**Description:** Queries the DC Correction mode setting for the FFT2 analyzer.

## DCCorrectionMode

**Command Syntax:** :Alyzr(i):FFT2Ch:DCCorrectionMode *Value* [, *AllowCoercion*]

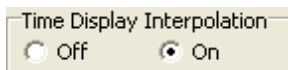
**Command Argument(s):** *Value* <int> {dcmNone=0 | dcmMean=1 | dcmPkPk=2}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:DCCorrectionMode dcmNone

**Related Command(s):** DCCorrectionMode?

**Description:** Sets the DC Correction mode setting for the FFT2 analyzer.



## DisplayInterpolation?

**Command Syntax:** :Alyzr(i):FFT2Ch:DisplayInterpolation?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:DisplayInterpolation] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:DisplayInterpolation?

**[ :Alyzr (0) :FFT2Ch:DisplayInterpolation ] False**

**Related Command(s):** DisplayInterpolation

**Description:** Queries the display interpolation mode for the FFT2 analyzer.

## DisplayInterpolation

**Command Syntax:** :Alyzr(i):FFT2Ch:DisplayInterpolation Value [, AllowCoercion]

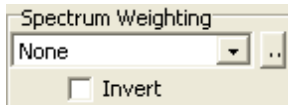
**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:DisplayInterpolation False

**Related Command(s):** DisplayInterpolation?

**Description:** Sets the display interpolation mode for the FFT2 analyzer.



## SetEqFile

**Command Syntax:** :Alyzr(i):FFT2Ch:SetEqFile FileName

**Command Argument(s):** FileName <string>

**Example:** :Alyzr (0) :FFT2Ch:SetEqFile AES1720.eq

**Description:** Specifies an EQ file to weight the FFT spectrum. The argument is the file name. In the absence of any path in the filename SR1 will look for the file in the user\eqCurves directory. To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e.  
:Alyzr (i) :FFT2ch:SetEqFile ""

## InvertEq?

**Command Syntax:** :Alyzr(i):FFT2Ch:InvertEq?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:InvertEq] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:InvertEq?  
[:Alyzr (0) :FFT2Ch:InvertEq] False

**Related Command(s):** InvertEq

**Description:** Queries the invert EQ status of the FFT2 analyzer. If on, the spectrum is weighted by the inverse of the response given by the EQ file.

## InvertEq

**Command Syntax:** :Alyzr(i):FFT2Ch:InvertEq Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:InvertEq False

**Related Command(s):** InvertEq?

**Description:** Sets the invert EQ status of the FFT2 analyzer. If on, the spectrum is weighted by the inverse of the response given by the EQ file.

Calc. Impulse Response

## CalcImpulseResponse?

**Command Syntax:** :Alyzr(i):FFT2Ch:CalcImpulseResponse?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:CalcImpulseResponse] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:CalcImpulseResponse?  
[:Alyzr (0) :FFT2Ch:CalcImpulseResponse] False

**Related Command(s):** CalcImpulseResponse

**Description:** Queries whether impulse response measurements will be calculated.

## CalcImpulseResponse

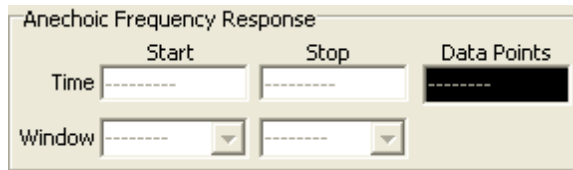
**Command Syntax:** :Alyzr(i):FFT2Ch:CalcImpulseResponse Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :FFT2Ch:CalcImpulseResponse False

**Related Command(s):** CalcImpulseResponse?

**Description:** Sets calculation of impulse response related measurements to on or off.



## IRStart?

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:IRStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:IRStart?  
[:Alyzr(0):FFT2Ch:IRStart] -0.016

**Related Command(s):** IRStart

**Description:** Queries the start time for the range of impulse response points used in the calculation of the anechoic frequency response. The values are relative to the midpoint of the time record.

## IRStart

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:IRStart -0.016 s

**Related Command(s):** IRStart?

**Description:** Sets the start time for the range of impulse response points used in the calculation of the anechoic frequency response. The values are relative to the midpoint of the time record.

## IRStop?

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:IRStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:IRStop?  
[:Alyzr(0):FFT2Ch:IRStop] 0.015984 s

**Related Command(s):** IRStop

**Description:** Queries the stop time for the range of impulse response points used in the calculation of the anechoic frequency response. The values are relative to the midpoint of the time record.



## IRStop

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:IRStop 0.015984 s

**Related Command(s):** IRStop?

**Description:** Sets the stop time for the range of impulse response points used in the calculation of the anechoic frequency response. The values are relative to the midpoint of the time record.

## IRStartWin?

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStartWin?

**Response Syntax:** [:Alyzr(i):FFT2Ch:IRStartWin] *Value*

**Response Argument(s):** *Value* <int> {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}

**Example:** :Alyzr(0):FFT2Ch:IRStartWin?  
 [:Alyzr(0):FFT2Ch:IRStartWin] irwNone

**Related Command(s):** IRStartWin

**Description:** Queries the window width for the raised cosine window used to transition at the beginning of the selected region of the impulse response. The width is given as a percentage of the total time record width.

## IRStartWin

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStartWin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:IRStartWin irwNone

**Related Command(s):** IRStartWin?

**Description:** Sets the window width for the raised cosine window used to transition at the beginning of the selected region of the impulse response. The width is given as a percentage of the total time record width.

## IRStopWin?

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStopWin?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:IRStopWin] *Value*

**Response Argument(s):** *Value* <int> {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}

**Example:** :Alyzr(0):FFT2Ch:IRStopWin?  
 [:Alyzr(0):FFT2Ch:IRStopWin] irwNone

**Related Command(s):** IRStopWin

**Description:** Queries the window width for the raised cosine window used to transition at the end of the selected region of the impulse response. The width is given as a percentage of the total time record width.

## IRStopWin

**Command Syntax:** :Alyzr(i):FFT2Ch:IRStopWin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):FFT2Ch:IRStopWin irwNone

**Related Command(s):** IRStopWin?

**Description:** Sets the window width for the raised cosine window used to transition at the end of the selected region of the impulse response. The width is given as a percentage of the total time record width.

## IRNumPoints?

**Command Syntax:** :Alyzr(i):FFT2Ch:IRNumPoints?

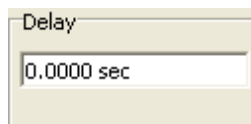
**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):FFT2Ch:IRNumPoints] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):FFT2Ch:IRNumPoints?  
 [:Alyzr(0):FFT2Ch:IRNumPoints] 500

**Description:** Queries the number of points in the region selected for use in calculating the anechoic frequency response.



## TriggerDelay?

**Command Syntax:** :Alyzr(i):FFT2Ch:TriggerDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):FFT2Ch:TriggerDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):FFT2Ch:TriggerDelay?  
 [:Alyzr(0):FFT2Ch:TriggerDelay] 0 s

**Related Command(s):** TriggerDelay

**Description:** Queries the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample (1/Fs). Negative (pre-triggered) delays are allowed.

## TriggerDelay

**Command Syntax:** :Alyzr(i):FFT2Ch:TriggerDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):FFT2Ch:TriggerDelay 0 s

**Related Command(s):** TriggerDelay?

**Description:** Sets the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample (1/Fs). Negative (pre-triggered) delays are allowed.

### Form Commands:

- :Alyzr(i):FFT2:OpenForm**
- :Alyzr(i):FFT2:OpenFormwID?**
- :Alyzr(i):FFT2:CloseForm**
- :Alyzr(i):FFT2:CloseForms**
- :Alyzr(i):FFT2:FormCount?**
- :Alyzr(i):FFT2:FormID?**

## 2.3.7.4 Time Domain Detector

<b>Object:</b>	<b>:Alyzr(i):TimeDomDetector</b> <b>:Alyzr(i):TDD</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands pertaining to the Time Domain Detector on either A0 or A1.



## Measurement?

**Command Syntax:** :Alyzr(i):TimeDomDetector:Measurement?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:Measurement] Value

**Response Argument(s):** Value <int> {adAmplitude=0 | adRatio=1 | adTHDNAmp=2 | adTHDNRatio=3 | adSINADRatio=4 | adCrestFactor=5}

**Example:** :Alyzr(0):TimeDomDetector:Measurement?

[ :Alyzr(0):TimeDomDetector:Measurement] adAmplitude

**Related Command(s):** Measurement

**Description:** Queries the measurement being performed by the TDD analyzer.

## Measurement

**Command Syntax:** :Alyzr(i):TimeDomDetector:Measurement Value [, AllowCoercion]

**Command Argument(s):** Value <int> {adAmplitude=0 | adRatio=1 | adTHDNAmp=2 | adTHDNRatio=3 | adSINADRatio=4 | adCrestFactor=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:Measurement adAmplitude

**Related Command(s):** Measurement?

**Description:** Sets the measurement being performed by the TDD analyzer.

## DetectorRdg?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:DetectorRdg? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:DetectorRdg] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):TimeDomDetector:DetectorRdg?  
[:Alyzr(0):TimeDomDetector:DetectorRdg] 1.00 Vrms`

**Description:** Queries the current measurement reading of the TDD.

## DetectRate?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:DetectRate?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:DetectRate] Value`

**Response Argument(s):** `Value <int> {adFast=0 | adPrecise=1 | adrps1=2 | adrps2=3 | adrps4=4 | adrps8=5 |  
adrps16=6 | adrps32=7 | adrps64=8 | adrps128=9 | adrps256=10}`

**Example:** `:Alyzr(0):TimeDomDetector:DetectRate?  
[:Alyzr(0):TimeDomDetector:DetectRate] adFast`

**Related Command(s):** DetectRate

**Description:** Queries the detector rate setting of the TDD analyzer.

## DetectRate

**Command Syntax:** `:Alyzr(i):TimeDomDetector:DetectRate Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {adFast=0 | adPrecise=1 | adrps1=2 | adrps2=3 | adrps4=4 | adrps8=5 |  
adrps16=6 | adrps32=7 | adrps64=8 | adrps128=9 | adrps256=10}  
AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):TimeDomDetector:DetectRate adFast`

**Related Command(s):** DetectRate?

**Description:** Sets the detector rate for the TDD analyzer.

## Response?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:Response?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:Response] Value`

**Response Argument(s):** `Value <int> {adRMS=0 | adQuasiPk=1 | adPeak=2}`

**Example:** `:Alyzr(0):TimeDomDetector:Response?  
[:Alyzr(0):TimeDomDetector:Response] adRMS`

**Related Command(s):** Response

**Description:** Queries the response setting of the TDD.

## Response

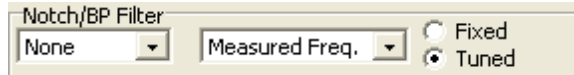
**Command Syntax:** :Alyzr(i):TimeDomDetector:Response *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {adRMS=0 | adQuasiPk=1 | adPeak=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:Response adRMS

**Related Command(s):** Response?

**Description:** Sets the response for the TDD.



## NotchBPFilt?

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPFilt?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:NotchBPFilt] *Value*

**Response Argument(s):** *Value* <int> {adNoNotchBP=0 | adNotch=1 | adBPHW=2 | adBP3rdOct=3 |  
adBP6thOct=4 | adBP12Oct=5 | adBP24Oct=6}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPFilt?

[ :Alyzr(0):TimeDomDetector:NotchBPFilt] adNoNotchBP

**Related Command(s):** NotchBPFilt

**Description:** Queries the Notch/Bandpass setting for the TDD analyzer.

## NotchBPFilt

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {adNoNotchBP=0 | adNotch=1 | adBPHW=2 | adBP3rdOct=3 |  
adBP6thOct=4 | adBP12Oct=5 | adBP24Oct=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPFilt adNoNotchBP

**Related Command(s):** NotchBPFilt?

**Description:** Sets the Notch/Bandpass setting for the TDD analyzer.

## NotchBPFixedFreq?

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPFixedFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):TimeDomDetector:NotchBPFixedFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):TimeDomDetector:NotchBPFixedFreq?

[ :Alyzr(0):TimeDomDetector:NotchBPFixedFreq] 1000 Hz

**Related Command(s):** NotchBPFixedFreq

**Description:** Queries the fixed notch/bandpass center frequency when the notch bandpass tuning mode is set to fixed frequency.

## NotchBPFixedFreq

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPFixedFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPFixedFreq 1000 Hz

**Related Command(s):** NotchBPFixedFreq?

**Description:** Sets the fixed notch/bandpass center frequency when the notch bandpass tuning mode is set to fixed frequency.

## NotchBPTuningMode?

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPTuningMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:NotchBPTuningMode] *Value*

**Response Argument(s):** *Value* <int> {tmFixed=0 | tmTuned=1}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPTuningMode?  
 [:Alyzr(0):TimeDomDetector:NotchBPTuningMode] tmFixed

**Related Command(s):** NotchBPTuningMode

**Description:** Queries the notch/bandpass tuning mode.

## NotchBPTuningMode

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPTuningMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {tmFixed=0 | tmTuned=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPTuningMode tmFixed

**Related Command(s):** NotchBPTuningMode?

**Description:** Sets the notch/bandpass tuning mode.

## NotchBPTuningSource?

**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPTuningSource?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:NotchBPTuningSource] *Value*

**Response Argument(s):** *Value* <int> {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 |  
 tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPTuningSource?  
 [:Alyzr(0):TimeDomDetector:NotchBPTuningSource] tsMeasFr

**Related Command(s):** NotchBPTuningSource

**Description:** Queries the notch/bandpass tuning source when the tuning mode is set to "tuned."

## NotchBPTuningSource

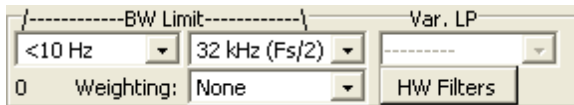
**Command Syntax:** :Alyzr(i):TimeDomDetector:NotchBPTuningSource *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:NotchBPTuningSource tsMeasFreq

**Related Command(s):** NotchBPTuningSource?

**Description:** Sets the notch/bandpass tuning source when the tuning mode is set to "tuned."



## BandEdgeLo?

**Command Syntax:** :Alyzr(i):TimeDomDetector:BandEdgeLo?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:BandEdgeLo] *Value*

**Response Argument(s):** *Value* <int> {adDC=0 | adHz22=1 | adHz100=2 | adHz400=3 | adSharp400=4 | adHzAES20=5 | adHzAES40=6 | adHzAES80=7}

**Example:** :Alyzr(0):TimeDomDetector:BandEdgeLo?

**[ :Alyzr(0):TimeDomDetector:BandEdgeLo ] adDC**

**Related Command(s):** BandEdgeLo

**Description:** Queries the setting of the low-frequency bandwidth limiting filter.

## BandEdgeLo

**Command Syntax:** :Alyzr(i):TimeDomDetector:BandEdgeLo *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {adDC=0 | adHz22=1 | adHz100=2 | adHz400=3 | adSharp400=4 | adHzAES20=5 | adHzAES40=6 | adHzAES80=7}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:BandEdgeLo adDC

**Related Command(s):** BandEdgeLo?

**Description:** Sets the low-frequency bandwidth limiting filter mode.



## BandEdgeHi?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:BandEdgeHi?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:BandEdgeHi] Value`

**Response Argument(s):** `Value <int> {adFsdiv2=0 | adHzVar=1 | adHzAES20k=2 | adHzAES40k=3 | adHzAES80k=4}`

**Example:** `:Alyzr(0):TimeDomDetector:BandEdgeHi?  
[:Alyzr(0):TimeDomDetector:BandEdgeHi] adFsdiv2`

**Related Command(s):** BandEdgeHi

**Description:** Queries the high-frequency bandwidth limiting filter mode.

## BandEdgeHi

**Command Syntax:** `:Alyzr(i):TimeDomDetector:BandEdgeHi Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {adFsdiv2=0 | adHzVar=1 | adHzAES20k=2 | adHzAES40k=3 | adHzAES80k=4}  
AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):TimeDomDetector:BandEdgeHi adFsdiv2`

**Related Command(s):** BandEdgeHi?

**Description:** Sets the high-frequency bandwidth limiting filter mode.

## BandEdgeHiVarFreq?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:BandEdgeHiVarFreq? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:BandEdgeHiVarFreq] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):TimeDomDetector:BandEdgeHiVarFreq?  
[:Alyzr(0):TimeDomDetector:BandEdgeHiVarFreq] 22000 Hz`

**Related Command(s):** BandEdgeHiVarFreq

**Description:** Queries the high-frequency filter cutoff when the high-frequency bandwidth limiting filter mode is set to "variable." (adHzVar)

## BandEdgeHiVarFreq

**Command Syntax:** `:Alyzr(i):TimeDomDetector:BandEdgeHiVarFreq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>  
AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):TimeDomDetector:BandEdgeHiVarFreq Value`

**Related Command(s):** BandEdgeHiVarFreq?

**Description:** Sets the high-frequency filter cutoff when the high-frequency bandwidth limiting filter mode is set to "variable." (adHzVar)

## WeightingFilt?

**Command Syntax:** :Alyzr(i):TimeDomDetector:WeightingFilt?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:WeightingFilt] *Value*

**Response Argument(s):** *Value* <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

**Example:** :Alyzr(0):TimeDomDetector:WeightingFilt?  
[:Alyzr(0):TimeDomDetector:WeightingFilt] adNoWt

**Related Command(s):** WeightingFilt

**Description:** Queries the weighting filter mode.

## WeightingFilt

**Command Syntax:** :Alyzr(i):TimeDomDetector:WeightingFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:WeightingFilt adNoWt

**Related Command(s):** WeightingFilt?

**Description:** Sets the weighting filter mode.



## PostFilterGain?

**Command Syntax:** :Alyzr(i):TimeDomDetector:PostFilterGain?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:PostFilterGain] *Value*

**Response Argument(s):** *Value* <int> {pfgAuto=0 | pfgdBminus12=1 | pfgdBminus6=2 | pfgdB0=3 | pfgdB6=4 | pfgdB12=5 | pfgdB18=6 | pfgdB24=7 | pfgdB30=8 | pfgdB36=9 | pfgdB42=10 | pfgdB48=11 | pfgdB54=12 | pfgdB60=13 | pfgdB66=14}

**Example:** :Alyzr(0):TimeDomDetector:PostFilterGain?  
[:Alyzr(0):TimeDomDetector:PostFilterGain] pfgAuto

**Related Command(s):** PostFilterGain

**Description:** Queries the post-filter gain setting. Note that post filter gain is only active for analog inputs when using the HiBandwidth converter.

## PostFilterGain

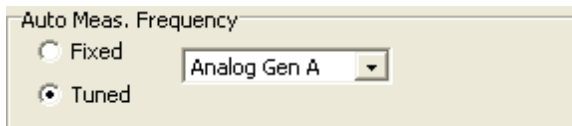
**Command Syntax:** `:Alyzr(i):TimeDomDetector:PostFilterGain Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {pfgAuto=0 | pfgdBminus12=1 | pfgdBminus6=2 | pfgdB0=3 | pfgdB6=4 | pfgdB12=5 | pfgdB18=6 | pfgdB24=7 | pfgdB30=8 | pfgdB36=9 | pfgdB42=10 | pfgdB48=11 | pfgdB54=12 | pfgdB60=13 | pfgdB66=14}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):TimeDomDetector:PostFilterGain pfgAuto`

**Related Command(s):** PostFilterGain?

**Description:** Sets the post-filter gain setting. Note that post filter gain is only active for analog inputs when using the HiBandwidth converter.



## AutoDetectMode?

**Command Syntax:** `:Alyzr(i):TimeDomDetector:AutoDetectMode?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):TimeDomDetector:AutoDetectMode] Value`

**Response Argument(s):** `Value <int> {tmFixed=0 | tmTuned=1}`

**Example:** `:Alyzr(0):TimeDomDetector:AutoDetectMode?`  
`[:Alyzr(0):TimeDomDetector:AutoDetectMode] tmFixed`

**Related Command(s):** AutoDetectMode

**Description:** Queries the fixed/tuned status of the TDD AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

## AutoDetectMode

**Command Syntax:** `:Alyzr(i):TimeDomDetector:AutoDetectMode Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {tmFixed=0 | tmTuned=1}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):TimeDomDetector:AutoDetectMode tmFixed`

**Related Command(s):** AutoDetectMode?

**Description:** Sets the fixed/tuned status of the TDD AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

## AutoDetect?

**Command Syntax:** :Alyzr(i):TimeDomDetector:AutoDetect?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):TimeDomDetector:AutoDetect] *Value*

**Response Argument(s):** *Value* <int> {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}

**Example:** :Alyzr(0):TimeDomDetector:AutoDetect?

**[ :Alyzr(0):TimeDomDetector:AutoDetect ] tsMeasFreq**

**Related Command(s):** AutoDetect

**Description:** Queries the source of the AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

## AutoDetect

**Command Syntax:** :Alyzr(i):TimeDomDetector:AutoDetect *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):TimeDomDetector:AutoDetect tsAnlgGenA

**Related Command(s):** AutoDetect?

**Description:** Sets the source of the TDD AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

## AutoDetectFreq?

**Command Syntax:** :Alyzr(i):TimeDomDetector:AutoDetectFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):TimeDomDetector:AutoDetectFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):TimeDomDetector:AutoDetectFreq?

**[ :Alyzr(0):TimeDomDetector:AutoDetectFreq ] 1000 Hz**

**Related Command(s):** AutoDetectFreq

**Description:** Queries the value of the manual AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

## AutoDetectFreq

**Command Syntax:** :Alyzr(i):TimeDomDetector:AutoDetectFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):TimeDomDetector:AutoDetectFreq 1000 Hz

**Related Command(s):** AutoDetectFreq?

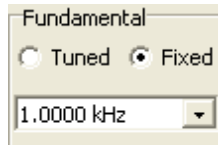
**Description:** Sets the value of the manual AutoDetect frequency. The AutoDetect frequency is used to determine the measurement interval; a lower frequency means a longer measurement interval.

### Form Commands:

**:Alyzr(i):TDD:OpenForm**  
**:Alyzr(i):TDD:OpenFormwID?**  
**:Alyzr(i):TDD:CloseForm**  
**:Alyzr(i):TDD:CloseForms**  
**:Alyzr(i):TDD:FormCount?**  
**:Alyzr(i):TDD:FormID?**

## 2.3.7.5 THD Analyzer

<b>Object:</b>	<b>:Alyzr(i):THD</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands pertaining to the THD analyzer on either A0 or A1.



## TuningMode?

**Command Syntax:** :Alyzr(i):THD:TuningMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:TuningMode] Value

**Response Argument(s):** Value <int> {tmFixed=0|tmTuned=1}

**Example:** :Alyzr(0):THD:TuningMode?

**[:Alyzr(0):THD:TuningMode] tmFixed**

**Related Command(s):** TuningMode

**Description:** Queries the fixed/tuned setting for the fundamental of the THD analyzer.

## TuningMode

**Command Syntax:** :Alyzr(i):THD:TuningMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {tmFixed=0|tmTuned=1}

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):THD:TuningMode tmFixed

**Related Command(s):** TuningMode?

**Description:** Sets the fixed/tuned setting for the fundamental of the THD analyzer.

## TuningSource?

**Command Syntax:** :Alyzr(i):THD:TuningSource?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:TuningSource] Value

**Response Argument(s):** Value <int> {tsMeasFreq=0|tsAnlgGenA=1|tsAnlgGenB=2|tsDigGenA=3|tsDigGenB=4|tsSweepSrc0=5|tsSweepSrc1=6}

**Example:** :Alyzr(0):THD:TuningSource?

**[:Alyzr(0):THD:TuningSource] tsMeasFreq**

**Related Command(s):** TuningSource

**Description:** Queries the fundamental tuning source when the tuning mode is "tuned."

## TuningSource

**Command Syntax:** :Alyzr(i):THD:TuningSource *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:TuningSource tsMeasFreq

**Related Command(s):** TuningSource?

**Description:** Sets the fundamental tuning source when the tuning mode is "tuned."

## FixedFreq?

**Command Syntax:** :Alyzr(i):THD:FixedFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):THD:FixedFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):THD:FixedFreq?  
 [:Alyzr(0):THD:FixedFreq] 1000 Hz

**Related Command(s):** FixedFreq

**Description:** Queries the fundamental frequency when the tuning mode is set to "fixed."

## FixedFreq

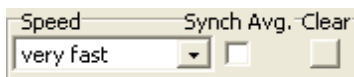
**Command Syntax:** :Alyzr(i):THD:FixedFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:FixedFreq 1000 Hz

**Related Command(s):** FixedFreq?

**Description:** Sets the fundamental frequency when the tuning mode is set to "fixed."



## Averaging?

**Command Syntax:** :Alyzr(i):THD:Averaging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:Averaging] *Value*

**Response Argument(s):** *Value* <int> {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}

**Example:** :Alyzr(0):THD:Averaging?  
 [:Alyzr(0):THD:Averaging] avgFast

**Related Command(s):** Averaging

**Description:** Queries the averaging mode for the THD analyzer.

## Averaging

**Command Syntax:** :Alyzr(i):THD:Averaging *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:Averaging avgFast

**Related Command(s):** Sets the averaging mode for the THD analyzer.

**Description:** Sets the averaging mode for the THD analyzer.

## SyncAveraging?

**Command Syntax:** :Alyzr(i):THD:SyncAveraging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:SyncAveraging] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):THD:SyncAveraging?  
 [:Alyzr(0):THD:SyncAveraging] 1

**Related Command(s):** SyncAveraging

**Description:** Queries the on/off status of synchronous averaging for the THD analyzer.

## SyncAveraging

**Command Syntax:** :Alyzr(i):THD:SyncAveraging *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:SyncAveraging 0

**Related Command(s):** SyncAveraging?

**Description:** Sets the on/off status of synchronous averaging for the THD analyzer.

## ResetAvg

**Command Syntax:** :Alyzr(i):THD:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):FFT:ResetAvg

**Description:** Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.





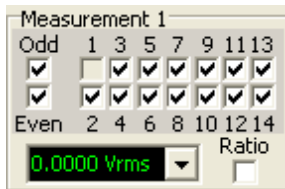
## SetEqFile

**Command Syntax:** :Alyzr(i):THD:SetEqFile *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Alyzr(0):THD:SetEqFile AWeighting.eq

**Description:** Sets an EQ file to weight the THD measurement. The argument is the file name, including the suffix. The default directory is "user\eqCurves". To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e.  
:Alyzr(i):FFT:SetEqFile ""



## Distortion0Rdg?

**Command Syntax:** :Alyzr(i):THD:Distortion0Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):THD:Distortion0Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):THD:Distortion0Rdg?

**[:Alyzr(0):THD:Distortion0Rdg] 0.00012 Vrms**

**Description:** Queries the "Measurement 1" reading of the THD analyzer. The value will be either an amplitude or a ratio depending on the setting of the "ratio" checkbox.

## Distortion1Rdg?

**Command Syntax:** :Alyzr(i):THD:Distortion1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):THD:Distortion1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):THD:Distortion1Rdg?

**[:Alyzr(0):THD:Distortion1Rdg] 0.0000023 %**

**Description:** Queries the "Measurement 2" reading of the THD analyzer. The value will be either an amplitude or a ratio depending on the setting of the "ratio" checkbox.

## MeasMode0?

**Command Syntax:** :Alyzr(i):THD:MeasMode0?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:MeasMode0] Value

**Response Argument(s):** Value <int> {thdAmplitude=0 | thdRatio=1}

**Example:** :Alyzr(0):THD:MeasMode0?

**[ :Alyzr(0):THD:MeasMode0 ] thdAmplitude**

**Related Command(s):** MeasMode0

**Description:** Queries the status of the ratio checkbox for measurement 1 of the THD analyzer. If on, the reported measurement is the ratio of the sum of the harmonic amplitudes to the fundamental amplitude. If off, the measurement is simply the sum of the selected harmonic amplitudes.

## MeasMode0

**Command Syntax:** :Alyzr(i):THD:MeasMode0 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {thdAmplitude=0 | thdRatio=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:MeasMode0 thdAmplitude

**Related Command(s):** MeasMode0?

**Description:** Sets the status of the ratio checkbox for measurement 1 of the THD analyzer. If on, the reported measurement is the ratio of the sum of the harmonic amplitudes to the fundamental amplitude. If off, the measurement is simply the sum of the selected harmonic amplitudes.

## MeasMode1?

**Command Syntax:** :Alyzr(i):THD:MeasMode1?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:MeasMode1] Value

**Response Argument(s):** Value <int> {thdAmplitude=0 | thdRatio=1}

**Example:** :Alyzr(0):THD:MeasMode1?

**[ :Alyzr(0):THD:MeasMode1 ] thdAmplitude**

**Related Command(s):** MeasMode1

**Description:** Queries the status of the ratio checkbox for measurement 2 of the THD analyzer. If on, the reported measurement is the ratio of the sum of the harmonic amplitudes to the fundamental amplitude. If off, the measurement is simply the sum of the selected harmonic amplitudes.

## MeasMode1

**Command Syntax:** :Alyzr(i):THD:MeasMode1 Value [, AllowCoercion]

**Command Argument(s):** Value <int> {thdAmplitude=0 | thdRatio=1}  
 AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:MeasMode1 thdAmplitude

**Related Command(s):** MeasMode1?

**Description:** Sets the status of the ratio checkbox for measurement 2 of the THD analyzer. If on, the reported measurement is the ratio of the sum of the harmonic amplitudes to the fundamental amplitude. If off, the measurement is simply the sum of the selected harmonic amplitudes.

## SelectedHarmonics0?

**Command Syntax:** :Alyzr(i):THD:SelectedHarmonics0?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:SelectedHarmonics0] Value

**Response Argument(s):** Value <int>

**Example:** :Alyzr(0):THD:SelectedHarmonics0?  
 [:Alyzr(0):THD:SelectedHarmonics0] 12

**Related Command(s):** SelectedHarmonics0

**Description:** Queries the harmonic selection for measurement 1 of the THD analyzer. The returned value is the sum of the values corresponding to the selected harmonics as follows:

2nd Harmonic =  $2^1 = 2$   
 3rd Harmonic =  $2^2 = 4$   
 4th Harmonic =  $2^3 = 8$   
 ...etc

## SelectedHarmonics0

**Command Syntax:** :Alyzr(i):THD:SelectedHarmonics0 Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
 AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):THD:SelectedHarmonics0 12

**Related Command(s):** SelectedHarmonics0?

**Description:** Sets the harmonic selection for measurement 1 of the THD analyzer. The argument is the sum of the values corresponding to the desired harmonics as follows:

2nd Harmonic =  $2^1 = 2$   
 3rd Harmonic =  $2^2 = 4$   
 4th Harmonic =  $2^3 = 8$   
 ...etc

## SelectedHarmonics1?

**Command Syntax:** :Alyzr(i):THD:SelectedHarmonics1?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):THD:SelectedHarmonics1] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):THD:SelectedHarmonics1?  
[:Alyzr(0):THD:SelectedHarmonics1] 12

**Related Command(s):** SelectedHarmonics1

**Description:** Queries the harmonic selection for measurement 2 of the THD analyzer. The returned value is the sum of the values corresponding to the selected harmonics as follows:

2nd Harmonic =  $2^1 = 2$

3rd Harmonic =  $2^2 = 4$

4th Harmonic =  $2^3 = 8$

...etc

## SelectedHarmonics1

**Command Syntax:** :Alyzr(i):THD:SelectedHarmonics1 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):THD:SelectedHarmonics1 12

**Related Command(s):** SelectedHarmonics1?

**Description:** Sets the harmonic selection for measurement 2 of the THD analyzer. The argument is the sum of the values corresponding to the desired harmonics as follows:

2nd Harmonic =  $2^1 = 2$

3rd Harmonic =  $2^2 = 4$

4th Harmonic =  $2^3 = 8$

...etc

## Form Commands:

**:Alyzr(i):THD:OpenForm**

**:Alyzr(i):THD:OpenFormwID?**

**:Alyzr(i):THD:CloseForm**

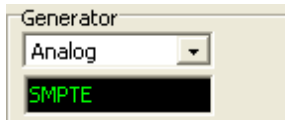
**:Alyzr(i):THD:CloseForms**

**:Alyzr(i):THD:FormCount?**

**:Alyzr(i):THD:FormID?**

2.3.7.6 IMD Analyzer

<b>Object:</b>	<b>:Alyzr(i):IMD</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0   1}
<i>Description:</i>	Commands pertaining to the IMD analyzer on either A0 or A1.



### GenLink?

**Command Syntax:** :Alyzr(i):IMD:GenLink?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):IMD:GenLink] *Value*

**Response Argument(s):** *Value* <int> {imdNoLink=-1 | imdDigGenA=0 | imdDigGenB=1 | imdAnlgGenA=2 | imdAnlgGenB=3}

**Example:** :Alyzr(0):IMD:GenLink?  
[:Alyzr(0):IMD:GenLink] imdNoLink

**Related Command(s):** GenLink

**Description:** Queries which generator will be used for the IMD measurement. The IMD analyzer determines whether to make an SMPTE/CCIF/DIM measurement based on this generator's settings.

### GenLink

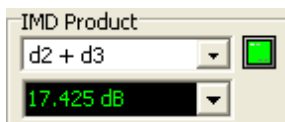
**Command Syntax:** :Alyzr(i):IMD:GenLink *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {imdNoLink=-1 | imdDigGenA=0 | imdDigGenB=1 | imdAnlgGenA=2 | imdAnlgGenB=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):IMD:GenLink imdAnlgGenA

**Related Command(s):** GenLink?

**Description:** Sets the generator whose settings will be used by the IMD analyzer.



## DistortionRdg?

**Command Syntax:** :Alyzr(i):IMD:DistortionRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):IMD:DistortionRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):IMD:DistortionRdg?  
[:Alyzr(0):IMD:DistortionRdg] -102.1 dB

**Description:** Queries the current distortion reading of the IMD analyzer.

## Product?

**Command Syntax:** :Alyzr(i):IMD:Product?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):IMD:Product] *Value*

**Response Argument(s):** *Value* <int> {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2}, {imdCCIF2=0 | imdCCIF23=1}, {imdDIMU5=0 | imdDIMU54=1}

**Example:** :Alyzr(0):IMD:Product?  
[:Alyzr(0):IMD:Product] imdSMPTE2

**Related Command(s):** Product

**Description:** Queries the IMD product selection for the IMD analyzer. Note the multiple enumerations due to the different IMD signals.

## Product

**Command Syntax:** :Alyzr(i):IMD:Product *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2}, {imdCCIF2=0 | imdCCIF23=1}, {imdDIMU5=0 | imdDIMU54=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):IMD:Product imdSMPTE2

**Related Command(s):** Product?

**Description:** Sets the IMD product selection for the IMD analyzer. Note the multiple enumerations due to the different IMD signals.

## IsValid?

**Command Syntax:** :Alyzr(i):IMD:IsValid?

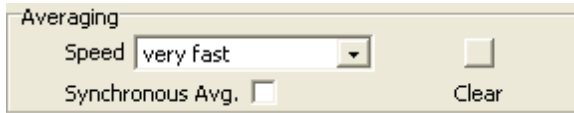
**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):IMD:IsValid] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):IMD:IsValid?  
[:Alyzr(0):IMD:IsValid] False

**Description:** Queries the current validity status of the IMD analyzer. The IMD measurement is valid if the selected distortion products lie within the analysis range of the currently selected converter and sampling rate.



## Averaging?

**Command Syntax:** :Alyzr(i):IMD:Averaging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):IMD:Averaging] Value

**Response Argument(s):** Value <int> {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}

**Example:** :Alyzr (0) :IMD:Averaging?

**[ :Alyzr (0) :IMD:Averaging ] avgFast**

**Related Command(s):** Averaging

**Description:** Queries the averaging selection for the IMD analyzer.

## Averaging

**Command Syntax:** :Alyzr(i):IMD:Averaging Value [, AllowCoercion]

**Command Argument(s):** Value <int> {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}  
 AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :IMD:Averaging avgFast

**Related Command(s):** Averaging?

**Description:** Sets the averaging selection for the IMD analyzer

## SyncAveraging?

**Command Syntax:** :Alyzr(i):IMD:SyncAveraging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):IMD:SyncAveraging] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Alyzr (0) :IMD:SyncAveraging?

**[ :Alyzr (0) :IMD:SyncAveraging ] False**

**Related Command(s):** SyncAveraging

**Description:** Queries the on/off status of synchronous averaging.

## SyncAveraging

**Command Syntax:** :Alyzr(i):IMD:SyncAveraging Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
 AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :IMD:SyncAveraging False

**Related Command(s):** SyncAveraging?

**Description:** Sets the on/off status of synchronous averaging.

## ResetAvg

**Command Syntax:** :Alyzr(i):IMD:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):IMD:ResetAvg

**Description:** Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



## SetEqFile

**Command Syntax:** :Alyzr(i):IMD:SetEqFile *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Alyzr(0):IMD:SetEqFileValue

**Description:** Sets an EQ file to weight the IMD measurement. The argument is the file name, including the suffix. The default directory is "user\eqCurves". To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e.  
:Alyzr(i):FFT:SetEqFile ""

## Form Commands:

**:Alyzr(i):IMD:OpenForm**

**:Alyzr(i):IMD:OpenFormwID?**

**:Alyzr(i):IMD:CloseForm**

**:Alyzr(i):IMD:CloseForms**

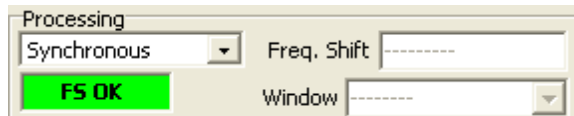
**:Alyzr(i):IMD:FormCount?**

**:Alyzr(i):IMD:FormID?**



2.3.7.7 Multitone Analyzer

<b>Object:</b>	<b>:Alyzr(i):Multitone</b> <b>:Alyzr(i):MTA</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the Multitone analyzer on either A0 or A1.



### Processing?

**Command Syntax:** :Alyzr(i):Multitone:Processing?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:Processing] Value

**Response Argument(s):** Value <int> {mpSync=0|mpWindow=1}

**Example:** :Alyzr(0):Multitone:Processing?

**[:Alyzr(0):Multitone:Processing] mpSync**

**Related Command(s):** Processing

**Description:** Queries the processing mode (Synchronous or Windowed) for the multitone analyzer.

### Processing

**Command Syntax:** :Alyzr(i):Multitone:Processing Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mpSync=0|mpWindow=1}

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:Processing mpSync

**Related Command(s):** Processing?

**Description:** Sets the processing mode (Synchronous or Windowed) for the multitone analyzer.

## FreqShift?

**Command Syntax:** :Alyzr(i):Multitone:FreqShift? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:FreqShift] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:FreqShift?  
[:Alyzr(0):Multitone:FreqShift] 4 %

**Related Command(s):** FreqShift

**Description:** Queries the frequency shift setting (the maximum tone shift through the DUT) for the multitone analyzer.

## FreqShift

**Command Syntax:** :Alyzr(i):Multitone:FreqShift *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:FreqShift 4%

**Related Command(s):** FreqShift?

**Description:** Sets the frequency shift setting (the maximum tone shift through the DUT) for the multitone analyzer.

## Window?

**Command Syntax:** :Alyzr(i):Multitone:Window?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:Window] *Value*

**Response Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}

**Example:** :Alyzr(0):Multitone:Window?  
[:Alyzr(0):Multitone:Window] fftBlackmanHarris

**Related Command(s):** Window

**Description:** Queries the window selection for the multitone analyzer running in windowed mode.

## Window

**Command Syntax:** :Alyzr(i):Multitone:Window *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:Window fftBlackmanHarris

**Related Command(s):** Window?

**Description:** Sets the window selection for the multitone analyzer running in windowed mode.



## MaxHarmonic?

**Command Syntax:** :Alyzr(i):Multitone:MaxHarmonic?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:MaxHarmonic] Value

**Response Argument(s):** Value <int>

**Example:** :Alyzr(0):Multitone:MaxHarmonic?  
[:Alyzr(0):Multitone:MaxHarmonic] 10

**Related Command(s):** MaxHarmonic

**Description:** Queries the maximum harmonic number analyzed by the multitone analyzer.

## MaxHarmonic

**Command Syntax:** :Alyzr(i):Multitone:MaxHarmonic Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:MaxHarmonic 10

**Related Command(s):** MaxHarmonic?

**Description:** Sets the maximum harmonic number analyzed by the multitone analyzer.

## MaxIMDPProduct?

**Command Syntax:** :Alyzr(i):Multitone:MaxIMDPProduct?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:MaxIMDPProduct] Value

**Response Argument(s):** Value <int>

**Example:** :Alyzr(0):Multitone:MaxIMDPProduct?  
[:Alyzr(0):Multitone:MaxIMDPProduct] 2

**Related Command(s):** MaxIMDPProduct

**Description:** Queries the order of the maximum IMD products computed by the multitone analyzer.

## MaxIMDPProduct

**Command Syntax:** :Alyzr(i):Multitone:MaxIMDPProduct Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:MaxIMDPProduct 2

**Related Command(s):** MaxIMDPProduct?

**Description:** Sets the order of the maximum IMD products computed by the multitone analyzer.

## MinHarmonic?

**Command Syntax:** :Alyzr(i):Multitone:MinHarmonic?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:MinHarmonic] Value

**Response Argument(s):** Value <int>

**Example:** :Alyzr(0):Multitone:MinHarmonic?  
[:Alyzr(0):Multitone:MinHarmonic] 2

**Related Command(s):** MinHarmonic

**Description:** Queries the minimum harmonic number analyzed by the multitone analyzer.

## MinHarmonic

**Command Syntax:** :Alyzr(i):Multitone:MinHarmonic Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:MinHarmonic 2

**Related Command(s):** MinHarmonic?

**Description:** Sets the minimum harmonic number analyzed by the multitone analyzer.



## SetEqFile

**Command Syntax:** :Alyzr(i):Multitone:SetEqFile FileName

**Command Argument(s):** FileName <string>

**Example:** :Alyzr(0):Multitone:SetEqFileValue AWeighting.eq

**Description:** Sets an EQ file to weight the multitone measurement. The argument is the file name, including the suffix. The default directory is "user\eqCurves". To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e.  
:Alyzr(i):Multitone:SetEqFile ""

## InvertEq?

**Command Syntax:** :Alyzr(i):Multitone:InvertEq?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:InvertEq] Value

**Response Argument(s):** Value <int> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:InvertEq?  
[:Alyzr(0):Multitone:InvertEq] False

**Related Command(s):** InvertEq

**Description:** Queries the invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.

## InvertEq

**Command Syntax:** `:Alyzr(i):Multitone:InvertEq Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Multitone:InvertEq False`

**Related Command(s):** InvertEq?

**Description:** Sets the invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.



## RelMode?

**Command Syntax:** `:Alyzr(i):Multitone:RelMode?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Multitone:RelMode] Value`

**Response Argument(s):** `Value <bool> {mtAbsolute=0 | mtRelGen=1 | mtRelGenOther=2 | mtRelTone=3 | mtRelToneOther=4 | mtRelTotal=5}`

**Example:** `:Alyzr(0):Multitone:RelMode?`  
`[:Alyzr(0):Multitone:RelMode] mtAbsolute`

**Related Command(s):** RelMode

**Description:** Queries the relative mode used in computing the scalar measurements for the multitone analyzer.

## RelMode

**Command Syntax:** `:Alyzr(i):Multitone:RelMode Value [, AllowCoercion]`

**Command Argument(s):** `Value <bool> {mtAbsolute=0 | mtRelGen=1 | mtRelGenOther=2 | mtRelTone=3 | mtRelToneOther=4 | mtRelTotal=5}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Multitone:RelMode mtAbsolute`

**Related Command(s):** RelMode?

**Description:** Sets the relative mode used in computing the scalar measurements for the multitone analyzer.

## AnalyzeNoise?

**Command Syntax:** :Alyzr(i):Multitone:AnalyzeNoise?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:AnalyzeNoise] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:AnalyzeNoise?

**[ :Alyzr(0):Multitone:AnalyzeNoise ] False**

**Related Command(s):** AnalyzeNoise

**Description:** Queries the status of the noise analysis checkbox. Noise analysis is only available in the synchronous processing mode.

## AnalyzeNoise

**Command Syntax:** :Alyzr(i):Multitone:AnalyzeNoise *Value* [, *AllowCoercion*]

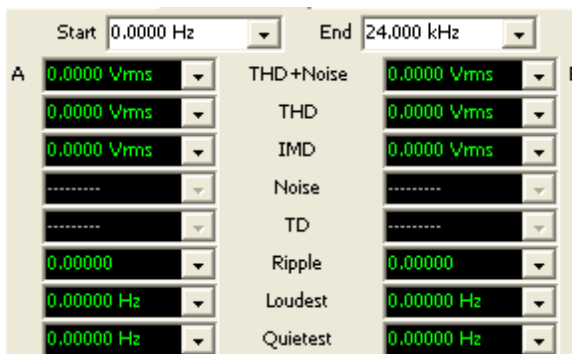
**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:AnalyzeNoise False

**Related Command(s):** AnalyzeNoise?

**Description:** Sets the status of the noise analysis checkbox. Noise analysis is only available in the synchronous processing mode.



## StartFreq?

**Command Syntax:** :Alyzr(i):Multitone:StartFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:StartFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:StartFreq?

**[ :Alyzr(0):Multitone:StartFreq ] 0 Hz**

**Related Command(s):** StartFreq

**Description:** Queries the minimum frequency used in the computation of the scalar measurements for the multitone analyzer.

## StartFreq

**Command Syntax:** :Alyzr(i):Multitone:StartFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:StartFreq 0 Hz

**Related Command(s):** StartFreq?

**Description:** Sets the minimum frequency used in the computation of the scalar measurements for the multitone analyzer.

## StopFreq?

**Command Syntax:** :Alyzr(i):Multitone:StopFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:StopFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:StopFreq?  
[:Alyzr(0):Multitone:StopFreq] 24000.0 Hz

**Related Command(s):** StopFreq

**Description:** Queries the maximum frequency used in the computation of the scalar measurements for the multitone analyzer.

## StopFreq

**Command Syntax:** :Alyzr(i):Multitone:StopFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:StopFreq 24000.0 Hz

**Related Command(s):** StopFreq?

**Description:** Sets the maximum frequency used in the computation of the scalar measurements for the multitone analyzer.

## THDNARdg?

**Command Syntax:** :Alyzr(i):Multitone:THDNARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:THDNARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:THDNARdg?  
[:Alyzr(0):Multitone:THDNARdg] 0.000012 %

**Description:** Queries the A channel THD<sub>N</sub> measurement.

## THDNBRdg?

**Command Syntax:** :Alyzr(i):Multitone:THDNBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:THDNBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:THDNBRdg?  
[:Alyzr(0):Multitone:THDNBRdg] 0.000012 %

**Description:** Queries the B channel THD+N measurement.

## THDARdg?

**Command Syntax:** :Alyzr(i):Multitone:THDARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:THDARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:THDARdg?  
[:Alyzr(0):Multitone:THDARdg] 0.000012 %

**Related Command(s):** THDARdg

**Description:** Queries the A channel THD+N measurement.

## THDARdg

**Command Syntax:** :Alyzr(i):Multitone:THDARdg *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Multitone:THDARdg 0.000012 %

**Related Command(s):** THDARdg?

**Description:** Queries the A channel THD measurement.

## THDBRdg?

**Command Syntax:** :Alyzr(i):Multitone:THDBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:THDBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:THDBRdg?  
[:Alyzr(0):Multitone:THDBRdg] 0.000012 %

**Related Command(s):** THDBRdg

**Description:** Queries the B channel THD measurement.



## IMDARdg?

**Command Syntax:** :Alyzr(i):Multitone:IMDARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:IMDARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:IMDARdg?  
[:Alyzr(0):Multitone:IMDARdg] -103.2 dB

**Related Command(s):** IMDARdg

**Description:** Queries the A channel IMD measurement.

## IMDBRdg?

**Command Syntax:** :Alyzr(i):Multitone:IMDBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:IMDBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:IMDBRdg?  
[:Alyzr(0):Multitone:IMDBRdg] -103.2 dB

**Related Command(s):** IMDBRdg

**Description:** Queries the B channel IMD measurement.

## NoiseARdg?

**Command Syntax:** :Alyzr(i):Multitone:NoiseARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:NoiseARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:NoiseARdg?  
[:Alyzr(0):Multitone:NoiseARdg] 0.00133 Vrms

**Description:** Queries the A channel noise measurement.

## NoiseBRdg?

**Command Syntax:** :Alyzr(i):Multitone:NoiseBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:NoiseBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:NoiseBRdg?  
[:Alyzr(0):Multitone:NoiseBRdg] 0.00133 Vrms

**Description:** Queries the B channel noise measurement.

## TotalDistARdg?

**Command Syntax:** :Alyzr(i):Multitone:TotalDistARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:TotalDistARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:TotalDistARdg?  
[:Alyzr(0):Multitone:TotalDistARdg] 0.0000159 %

**Description:** Queries the A channel total distortion measurement.

## TotalDistBRdg?

**Command Syntax:** :Alyzr(i):Multitone:TotalDistBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:TotalDistBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:TotalDistBRdg?  
[:Alyzr(0):Multitone:TotalDistBRdg] 0.0000159 %

**Description:** Queries the B channel total distortion measurement.

## RippleARdg?

**Command Syntax:** :Alyzr(i):Multitone:RippleARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:RippleARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:RippleARdg?  
[:Alyzr(0):Multitone:RippleARdg] 1.5 %

**Description:** Queries the A channel total ripple measurement.

## RippleBRdg?

**Command Syntax:** :Alyzr(i):Multitone:RippleBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:RippleBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:RippleBRdg?  
[:Alyzr(0):Multitone:RippleBRdg] 1.5 %

**Description:** Queries the B channel ripple measurement.

## FreqMaxARdg?

**Command Syntax:** :Alyzr(i):Multitone:FreqMaxARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:FreqMaxARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:FreqMaxARdg?  
[:Alyzr(0):Multitone:FreqMaxARdg] 10000 Hz

**Description:** Queries the A channel loudest tone frequency..

## FreqMaxBRdg?

**Command Syntax:** :Alyzr(i):Multitone:FreqMaxBRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:FreqMaxBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:FreqMaxBRdg?  
[:Alyzr(0):Multitone:FreqMaxBRdg] 10000 Hz

**Description:** Queries the B channel loudest tone frequency.

## FreqMinARdg?

**Command Syntax:** :Alyzr(i):Multitone:FreqMinARdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:FreqMinARdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:FreqMinARdg?  
[:Alyzr(0):Multitone:FreqMinARdg] 10000 Hz

**Description:** Queries the A channel quietest tone frequency.

## FreqMinBRdg?

**Command Syntax:** :Alyzr(i):Multitone:FreqMinBRdg? [*ValueUnit*]

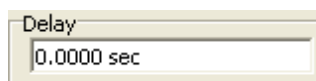
**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:FreqMinBRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:FreqMinBRdg?  
[:Alyzr(0):Multitone:FreqMinBRdg] 10000 Hz

**Description:** Queries the B channel quietest tone frequency.



## TriggerDelay?

**Command Syntax:** :Alyzr(i):Multitone:TriggerDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Multitone:TriggerDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Multitone:TriggerDelay?

**[ :Alyzr(0):Multitone:TriggerDelay] Value**

**Related Command(s):** TriggerDelay

**Description:** Queries the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample, (1/Fs). Negative (pre-trigger) delays are allowed.

## TriggerDelay

**Command Syntax:** :Alyzr(i):Multitone:TriggerDelay *Value* [, *AllowCoercion*]

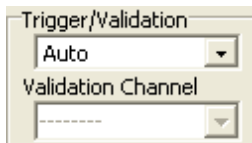
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:TriggerDelay *Value*

**Related Command(s):** TriggerDelay?

**Description:** Sets the delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample, (1/Fs). Negative (pre-trigger) delays are allowed.



## TriggerMode?

**Command Syntax:** :Alyzr(i):Multitone:TriggerMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:TriggerMode] *Value*

**Response Argument(s):** *Value* <int> {mttAuto=0 | mttTrigger=1 | mttValLoose=2 | mttValNorm=3 | mttValTight=4}

**Example:** :Alyzr(0):Multitone:TriggerMode?

**[ :Alyzr(0):Multitone:TriggerMode] mttAuto**

**Related Command(s):** TriggerMode

**Description:** Queries the trigger/validation mode for the multitone analyzer.

## TriggerMode

**Command Syntax:** :Alyzr(i):Multitone:TriggerMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {mttAuto=0 | mttTrigger=1 | mttValLoose=2 | mttValNorm=3 | mttValTight=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:TriggerMode mttAuto

**Related Command(s):** TriggerMode?

**Description:** Sets the trigger/validation mode for the multitone analyzer.

## ValidationChannel?

**Command Syntax:** :Alyzr(i):Multitone:ValidationChannel?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Multitone:ValidationChannel] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):Multitone:ValidationChannel?  
 [:Alyzr(0):Multitone:ValidationChannel] 0

**Related Command(s):** ValidationChannel

**Description:** Queries the channel used for time-record validation.

## ValidationChannel

**Command Syntax:** :Alyzr(i):Multitone:ValidationChannel *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Multitone:ValidationChannel 0

**Related Command(s):** ValidationChannel?

**Description:** Sets the channel used for time-record validation.

## Form Commands:

**:Alyzr(i):MTA:OpenForm**

**:Alyzr(i):MTA:OpenFormwID?**

**:Alyzr(i):MTA:CloseForm**

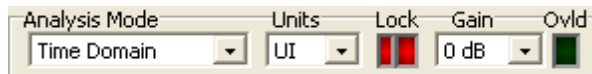
**:Alyzr(i):MTA:CloseForms**

**:Alyzr(i):MTA:FormCount?**

**:Alyzr(i):MTA:FormID?**

## 2.3.7.8 Jitter Analyzer

<b>Object:</b>	<b>:Alyzr(i):Jitter</b> <b>:Alyzr(i):JITT</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands pertaining to the Jitter Analyzer

**Domain?**

**Command Syntax:** :Alyzr(i):Jitter:Domain?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:Domain] Value

**Response Argument(s):** Value <int> {jdTime=0|jdFreq=1}

**Example:** :Alyzr(0):Jitter:Domain?

**[:Alyzr(0):Jitter:Domain] jdTime**

**Related Command(s):** Domain

**Description:** Queries the domain, time or frequency, for the jitter analyzer.

**Domain**

**Command Syntax:** :Alyzr(i):Jitter:Domain Value [, AllowCoercion]

**Command Argument(s):** Value <int> {jdTime=0|jdFreq=1}

AllowCoercion <bool> {False=0|True=1}

**Example:** :Alyzr(0):Jitter:Domain jdTime

**Related Command(s):** Domain?

**Description:** Sets the domain, time or frequency, for the jitter analyzer.

**MeasUnits?**

**Command Syntax:** :Alyzr(i):Jitter:MeasUnits?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:MeasUnits] Value

**Response Argument(s):** Value <int> {muUI=0|muSec=1}

**Example:** :Alyzr(0):Jitter:MeasUnits?

**[:Alyzr(0):Jitter:MeasUnits] muUI**

**Related Command(s):** MeasUnits

**Description:** Queries the jitter units that will be used by the analyzer..

## MeasUnits

**Command Syntax:** `:Alyzr(i):Jitter:MeasUnits Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {muUI=0 | muSec=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:MeasUnits muUI`

**Related Command(s):** MeasUnits?

**Description:** Sets the jitter units that will be used by the analyzer.

## ClockLockRdg?

**Command Syntax:** `:Alyzr(i):Jitter:ClockLockRdg?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:ClockLockRdg] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:ClockLockRdg?`  
`[ :Alyzr(0) :Jitter:ClockLockRdg] False`

**Description:** Queries the lock status of the digital audio receiver.

## DemodLockRdg?

**Command Syntax:** `:Alyzr(i):Jitter:DemodLockRdg?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:DemodLockRdg] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:DemodLockRdg?`  
`[ :Alyzr(0) :Jitter:DemodLockRdg] False`

**Description:** Queries the lock status of the jitter demodulator.

## ADCOverloadRdg?

**Command Syntax:** `:Alyzr(i):Jitter:ADCOverloadRdg?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:ADCOverloadRdg] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:ADCOverloadRdg?`  
`[ :Alyzr(0) :Jitter:ADCOverloadRdg] False`

**Description:** Queries the overload status of the jitter analyzer.

## Gain?

**Command Syntax:** :Alyzr(i):Jitter:Gain?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:Gain] Value

**Response Argument(s):** Value <int> {jg1x=0 | jg10x=1 | jg20x=2 | jg30x=3}

**Example:** :Alyzr(0):Jitter:Gain?

**[ :Alyzr(0):Jitter:Gain ] jg1x**

**Related Command(s):** Gain

**Description:** Queries the jitter analyzer gain setting.

## Gain

**Command Syntax:** :Alyzr(i):Jitter:Gain Value [, AllowCoercion]

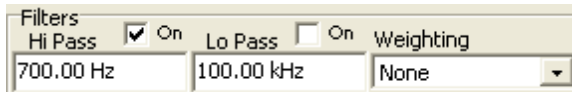
**Command Argument(s):** Value <int> {jg1x=0 | jg10x=1 | jg20x=2 | jg30x=3}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:Gain jg1x

**Related Command(s):** Gain?

**Description:** Sets the jitter analyzer gain.



## Time Weighting Filt?

**Command Syntax:** :Alyzr(i):Jitter:TimeWeightingFilt?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:TimeWeightingFilt] Value

**Response Argument(s):** Value <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

**Example:** :Alyzr(0):Jitter:TimeWeightingFilt?

**[ :Alyzr(0):Jitter:TimeWeightingFilt ] adNoWt**

**Related Command(s):** TimeWeightingFilt

**Description:** Queries the weighting filter used by the jitter analyzer in time domain mode.



## TimeWeightingFilt

**Command Syntax:** `:Alyzr(i):Jitter:TimeWeightingFilt Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):Jitter:TimeWeightingFilt adNoWt`

**Related Command(s):** TimeWeightingFilt?

**Description:** Sets the weighting filter used by the jitter analyzer in time domain mode.

## BandEdgeHi?

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeHi? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Jitter:BandEdgeHi] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Jitter:BandEdgeHi?`  
`[:Alyzr(0):Jitter:BandEdgeHi] Value`

**Related Command(s):** BandEdgeHi

**Description:** Queries the high frequency band-limiting filter frequency for time-domain mode.

## BandEdgeHi

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeHi Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
**AllowCoercion <bool> {False=0 | True=1}**

**Example:** `:Alyzr(0):Jitter:BandEdgeHi Value`

**Related Command(s):** BandEdgeHi?

**Description:** Sets the high frequency band-limiting filter frequency for time-domain mode.

## BandEdgeHiOn?

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeHiOn?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:BandEdgeHiOn] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:BandEdgeHiOn?`  
`[:Alyzr(0):Jitter:BandEdgeHiOn] False`

**Related Command(s):** BandEdgeHiOn

**Description:** Queries the on/off status of the time-domain high frequency band-limiting filter.

## BandEdgeHiOn

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeHiOn Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:BandEdgeHiOn False`

**Related Command(s):** BandEdgeHiOn?

**Description:** Sets the on/off status of the time-domain high frequency band-limiting filter.

## BandEdgeLo?

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeLo? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Jitter:BandEdgeLo] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Jitter:BandEdgeLo?`  
`[:Alyzr(0):Jitter:BandEdgeLo] Value`

**Related Command(s):** BandEdgeLo

**Description:** Queries the low frequency band-limiting filter frequency for time-domain mode.

## BandEdgeLo

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeLo Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:BandEdgeLo Value`

**Related Command(s):** BandEdgeLo?

**Description:** Sets the low frequency band-limiting filter frequency for time-domain mode.

## BandEdgeLoOn?

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeLoOn?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:BandEdgeLoOn] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:BandEdgeLoOn?`  
`[:Alyzr(0):Jitter:BandEdgeLoOn] False`

**Related Command(s):** BandEdgeLoOn

**Description:** Queries the on/off status of the time-domain low frequency band-limiting filter.

## BandEdgeLoOn

**Command Syntax:** `:Alyzr(i):Jitter:BandEdgeLoOn Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:BandEdgeLoOn False`

**Related Command(s):** BandEdgeLoOn?

**Description:** Sets the on/off status of the time-domain low frequency band-limiting filter

## FreqSetEqFile

**Command Syntax:** `:Alyzr(i):Jitter:FreqSetEqFile FileName`

**Command Argument(s):** `FileName <string>`

**Example:** `:Alyzr(0):Jitter:FreqSetEqFile Value`

**Description:** Sets the frequency-domain weighting filter.

## InvertEq?

**Command Syntax:** `:Alyzr(i):Jitter:InvertEq?`

**Command Argument(s):**

**Response Syntax:** `[:Alyzr(i):Jitter:InvertEq] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:InvertEq?`  
`[:Alyzr(0):Jitter:InvertEq] False`

**Related Command(s):** InvertEq

**Description:** Queries the invert/non-invert status of the frequency-domain weighting filter.

## InvertEq

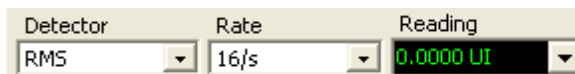
**Command Syntax:** `:Alyzr(i):Jitter:InvertEq Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Jitter:InvertEq False`

**Related Command(s):** InvertEq?

**Description:** Sets the invert/non-invert status of the frequency-domain weighting filter.



## TimeResponse?

**Command Syntax:** :Alyzr(i):Jitter:TimeResponse?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:TimeResponse] Value

**Response Argument(s):** Value <int> {jtRMS=0|jtPeak=1}

**Example:** :Alyzr(0):Jitter:TimeResponse?

**[ :Alyzr(0):Jitter:TimeResponse ] jtRMS**

**Related Command(s):** TimeResponse

**Description:** Queries the time-domain response type (rms or peak) of the jitter analyzer.

## TimeResponse

**Command Syntax:** :Alyzr(i):Jitter:TimeResponse Value [, AllowCoercion]

**Command Argument(s):** Value <int> {jtRMS=0|jtPeak=1}

**AllowCoercion** <bool> {False=0|True=1}

**Example:** :Alyzr(0):Jitter:TimeResponse jtRMS

**Related Command(s):** TimeResponse?

**Description:** Sets the time-domain response type of the jitter analyzer.

## TimeDetectRate?

**Command Syntax:** :Alyzr(i):Jitter:TimeDetectRate?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:TimeDetectRate] Value

**Response Argument(s):** Value <int> {jtDigFs=0|jtJitGen=1|jtrps1=2|jtrps2=3|jtrps4=4|jtrps8=5|jtrps16=6|jtrps32=7|jtrps64=8|jtrps128=9|jtrps256=10}

**Example:** :Alyzr(0):Jitter:TimeDetectRate?

**[ :Alyzr(0):Jitter:TimeDetectRate ] jtDigFs**

**Related Command(s):** TimeDetectRate

**Description:** Queries the time-domain measurement rate.

## TimeDetectRate

**Command Syntax:** :Alyzr(i):Jitter:TimeDetectRate Value [, AllowCoercion]

**Command Argument(s):** Value <int> {jtDigFs=0|jtJitGen=1|jtrps1=2|jtrps2=3|jtrps4=4|jtrps8=5|jtrps16=6|jtrps32=7|jtrps64=8|jtrps128=9|jtrps256=10}

**AllowCoercion** <bool> {False=0|True=1}

**Example:** :Alyzr(0):Jitter:TimeDetectRate jtDigFs

**Related Command(s):** TimeDetectRate?

**Description:** Sets the time-domain measurement rate.

## TimeJitterRdg?

**Command Syntax:** :Alyzr(i):Jitter:TimeJitterRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

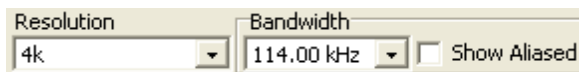
**Response Syntax:** [:Alyzr(i):Jitter:TimeJitterRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:TimeJitterRdg?

**[ :Alyzr(0) :Jitter:TimeJitterRdg] Value**

**Description:** Queries the jitter measured by the time-domain jitter analyzer.



## FreqLines?

**Command Syntax:** :Alyzr(i):Jitter:FreqLines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqLines] *Value*

**Response Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

**Example:** :Alyzr(0):Jitter:FreqLines?

**[ :Alyzr(0) :Jitter:FreqLines] fft132k**

**Related Command(s):** FreqLines

**Description:** Queries the number of FFT lines used by the frequency-domain jitter analyzer.

## FreqLines

**Command Syntax:** :Alyzr(i):Jitter:FreqLines *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqLines fft132k

**Related Command(s):** FreqLines?

**Description:** Sets the number of FFT lines used by the frequency-domain jitter analyzer.

## FreqNumSpansDown?

**Command Syntax:** :Alyzr(i):Jitter:FreqNumSpansDown?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqNumSpansDown] *Value*

**Response Argument(s):** *Value* <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 |  
fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}

**Example:** :Alyzr(0):Jitter:FreqNumSpansDown?

**[ :Alyzr(0) :Jitter:FreqNumSpansDown ] fftFsDiv2**

**Related Command(s):** FreqNumSpansDown

**Description:** Queries the frequency range for the frequency-domain jitter analyzer. The largest frequency range corresponds to  $F_s/2$ , each subsequent range is  $\times 2$  smaller.

## FreqNumSpansDown

**Command Syntax:** :Alyzr(i):Jitter:FreqNumSpansDown *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 |  
fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqNumSpansDown fftFsDiv2

**Related Command(s):** FreqNumSpansDown?

**Description:** Sets the frequency range for the frequency-domain jitter analyzer. The largest frequency range corresponds to  $F_s/2$ , each subsequent range is  $\times 2$  smaller.

## FreqTimeRecordDuration?

**Command Syntax:** :Alyzr(i):Jitter:FreqTimeRecordDuration? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqTimeRecordDuration] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqTimeRecordDuration?

**[ :Alyzr(0) :Jitter:FreqTimeRecordDuration ] .032 s**

**Description:** Queries the jitter time-record duration corresponding to the currently set frequency span.

## FreqShowAllLines?

**Command Syntax:** :Alyzr(i):Jitter:FreqShowAllLines?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqShowAllLines] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqShowAllLines?

**[ :Alyzr(0) :Jitter:FreqShowAllLines ] False**

**Related Command(s):** FreqShowAllLines

**Description:** Queries the on off status of "Show All Lines" for the frequency-domain jitter analyzer. When on, the jitter analyzer displays all lines from DC to  $F_s/2$ . When off, the analyzer displays lines out to a frequency guaranteed to be alias protected by at least 100 dB.

## FreqShowAllLines

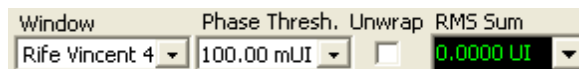
**Command Syntax:** :Alyzr(i):Jitter:FreqShowAllLines *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqShowAllLines False

**Related Command(s):** FreqShowAllLines?

**Description:** Sets the on off status of "Show All Lines" for the frequency-domain jitter analyzer. When on, the jitter analyzer displays all lines from DC to Fs/2. When off, the analyzer displays lines out to a frequency guaranteed to be alias protected by at least 100 dB.



## FreqWindow?

**Command Syntax:** :Alyzr(i):Jitter:FreqWindow?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqWindow] *Value*

**Response Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}

**Example:** :Alyzr(0):Jitter:FreqWindow?

**[:Alyzr(0):Jitter:FreqWindow] fftBlackmanHarris**

**Related Command(s):** FreqWindow

**Description:** Queries the window used by frequency-domain jitter analyzer.

## FreqWindow

**Command Syntax:** :Alyzr(i):Jitter:FreqWindow *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fftBlackmanHarris=0 | fftHann=1 | fftHamming=2 | fftEquiripple=3 |  
fftFlattop=4 | fftGaussian=5 | fftKaiser=6 | fftUniform=7 | fftRifeVincent4=8 |  
fftRifeVincent5=9 | fftRifeVincent10=10 | fftBlackmanHarris7=11}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqWindow fftBlackmanHarris

**Related Command(s):** FreqWindow?

**Description:** Sets the window used by frequency-domain jitter analyzer.

## FreqPhaseThreshold?

**Command Syntax:** :Alyzr(i):Jitter:FreqPhaseThreshold? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqPhaseThreshold] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqPhaseThreshold?  
[:Alyzr(0):Jitter:FreqPhaseThreshold] **Value**

**Related Command(s):** FreqPhaseThreshold

**Description:** Queries the amplitude threshold for calculating jitter phase.

## FreqPhaseThreshold

**Command Syntax:** :Alyzr(i):Jitter:FreqPhaseThreshold *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqPhaseThreshold *Value*

**Related Command(s):** FreqPhaseThreshold?

**Description:** Sets the amplitude threshold for calculating jitter phase.

## FreqPhaseUnwrap?

**Command Syntax:** :Alyzr(i):Jitter:FreqPhaseUnwrap?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqPhaseUnwrap] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqPhaseUnwrap?  
[:Alyzr(0):Jitter:FreqPhaseUnwrap] **False**

**Related Command(s):** FreqPhaseUnwrap

**Description:** Queries the on/off status of jitter phase unwrap.

## FreqPhaseUnwrap

**Command Syntax:** :Alyzr(i):Jitter:FreqPhaseUnwrap *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqPhaseUnwrap **False**

**Related Command(s):** FreqPhaseUnwrap?

**Description:** Sets the on/off status of jitter phase unwrap.



## FreqRmsJitter?

**Command Syntax:** :Alyzr(i):Jitter:FreqRmsJitter? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

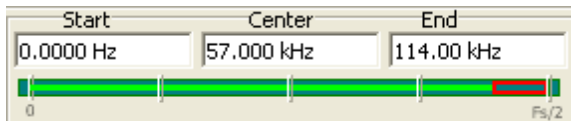
**Response Syntax:** [:Alyzr(i):Jitter:FreqRmsJitter] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqRmsJitter?

**[ :Alyzr(0):Jitter:FreqRmsJitter ] Value**

**Description:** Queries the total jitter calculated by the frequency domain jitter analyzer.



## FreqStart?

**Command Syntax:** :Alyzr(i):Jitter:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqStart?

**[ :Alyzr(0):Jitter:FreqStart ] 0.0**

**Related Command(s):** FreqStart

**Description:** Queries the start frequency of the analysis range for the frequency-domain jitter analyzer.

## FreqStart

**Command Syntax:** :Alyzr(i):Jitter:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Jitter:FreqStart 1000 Hz

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency of the analysis range for the frequency-domain jitter analyzer.

## FreqCenter?

**Command Syntax:** :Alyzr(i):Jitter:FreqCenter? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqCenter] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqCenter?  
[:Alyzr(0):Jitter:FreqCenter] Value

**Related Command(s):** FreqCenter

**Description:** Queries the center frequency of the analysis range for the frequency-domain jitter analyzer.

## FreqCenter

**Command Syntax:** :Alyzr(i):Jitter:FreqCenter *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqCenter Value

**Related Command(s):** FreqCenter?

**Description:** Sets the center frequency of the analysis range for the frequency-domain jitter analyzer.

## FreqStop?

**Command Syntax:** :Alyzr(i):Jitter:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqStop?  
[:Alyzr(0):Jitter:FreqStop] Value

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency of the analysis range for the frequency-domain jitter analyzer

## FreqStop

**Command Syntax:** :Alyzr(i):Jitter:FreqStop *Value* [, *AllowCoercion*]

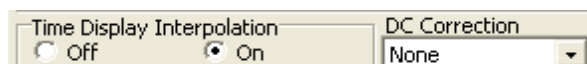
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqStop Value

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency of the analysis range for the frequency-domain jitter analyzer



## FreqDisplayInterpolation?

**Command Syntax:** :Alyzr(i):Jitter:FreqDisplayInterpolation?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqDisplayInterpolation] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr (0) :Jitter:FreqDisplayInterpolation?  
[:Alyzr (0) :Jitter:FreqDisplayInterpolation] False

**Related Command(s):** FreqDisplayInterpolation

**Description:** Queries the on/off status of display interpolation for the frequency-domain jitter analyzer.

## FreqDisplayInterpolation

**Command Syntax:** :Alyzr(i):Jitter:FreqDisplayInterpolation *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :Jitter:FreqDisplayInterpolation False

**Related Command(s):** FreqDisplayInterpolation?

**Description:** Sets the on/off status of display interpolation for the frequency-domain jitter analyzer.

## FreqDCCorrectionMode?

**Command Syntax:** :Alyzr(i):Jitter:FreqDCCorrectionMode?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqDCCorrectionMode] *Value*

**Response Argument(s):** *Value* <int> {jdcNone=0 | jdcAvg=1 | jdcHalfPkPk=2}

**Example:** :Alyzr (0) :Jitter:FreqDCCorrectionMode?  
[:Alyzr (0) :Jitter:FreqDCCorrectionMode] jdcNone

**Related Command(s):** FreqDCCorrectionMode

**Description:** Queries the DC correction mode for the frequency-domain jitter analyzer.

## FreqDCCorrectionMode

**Command Syntax:** :Alyzr(i):Jitter:FreqDCCorrectionMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {jdcNone=0 | jdcAvg=1 | jdcHalfPkPk=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr (0) :Jitter:FreqDCCorrectionMode jdcNone

**Related Command(s):** FreqDCCorrectionMode?

**Description:** Sets the DC correction mode for the frequency-domain jitter analyzer.



## FreqAveraging?

**Command Syntax:** :Alyzr(i):Jitter:FreqAveraging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqAveraging] *Value*

**Response Argument(s):** *Value* <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2 | avgPeakFixedLength=3 | avgPeakContinuous=4}

**Example:** :Alyzr(0):Jitter:FreqAveraging?

**[ :Alyzr(0):Jitter:FreqAveraging ] avgNone**

**Related Command(s):** FreqAveraging

**Description:** Queries the averaging mode for the frequency domain jitter analyzer.

## FreqAveraging

**Command Syntax:** :Alyzr(i):Jitter:FreqAveraging *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {avgNone=0 | avgFixedLength=1 | avgContinuous=2 | avgPeakFixedLength=3 | avgPeakContinuous=4}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqAveraging avgNone

**Related Command(s):** FreqAveraging?

**Description:** Sets the averaging mode for the frequency domain jitter analyzer..

## FreqAvgDone?

**Command Syntax:** :Alyzr(i):Jitter:FreqAvgDone?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqAvgDone] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqAvgDone?

**[ :Alyzr(0):Jitter:FreqAvgDone ] False**

**Description:** Queries the "done" status for frequency-domain jitter averaging.

## FreqNumAverages?

**Command Syntax:** :Alyzr(i):Jitter:FreqNumAverages?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Jitter:FreqNumAverages] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):Jitter:FreqNumAverages?

**[ :Alyzr(0):Jitter:FreqNumAverages ] 10**

**Related Command(s):** FreqNumAverages

**Description:** Queries the number of averages performed the analyzer.

## FreqNumAverages

**Command Syntax:** :Alyzr(i):Jitter:FreqNumAverages *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqNumAverages 10

**Related Command(s):** FreqNumAverages?

**Description:** Sets the number of averages performed the analyzer.

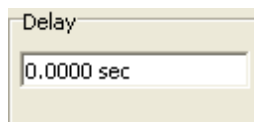
## ResetAvg

**Command Syntax:** :Alyzr(i):Jitter:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):Jitter:ResetAvg

**Description:** Resets the average buffer of the analyzer.



## FreqTriggerDelay?

**Command Syntax:** :Alyzr(i):Jitter:FreqTriggerDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Jitter:FreqTriggerDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Jitter:FreqTriggerDelay?  
 [:Alyzr(0):Jitter:FreqTriggerDelay] 0.016 s

**Related Command(s):** FreqTriggerDelay

**Description:** Queries the delay between the trigger and the start of the time record for the frequency-domain jitter analyzer.

## FreqTriggerDelay

**Command Syntax:** :Alyzr(i):Jitter:FreqTriggerDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Jitter:FreqTriggerDelay 0.016 s

**Related Command(s):** FreqTriggerDelay?

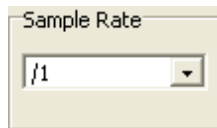
**Description:** Sets the delay between the trigger and the start of the time record for the frequency-domain jitter analyzer.

### *Supported Form Commands:*

**:Alyzr(i):JITT:OpenForm**  
**:Alyzr(i):JITT:OpenFormwID?**  
**:Alyzr(i):JITT:CloseForm**  
**:Alyzr(i):JITT:CloseForms**  
**:Alyzr(i):JITT:FormCount?**  
**:Alyzr(i):JITT:FormID?**

2.3.7.9 Histogram Analyzer

<b>Object:</b>	<b>:Alyzr(i):Histogram</b> <b>:Alyzr(i):HIST</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0 1}
<i>Description:</i>	Commands related to the histogram analyzer on A0 or A1.



### FsDivider?

**Command Syntax:** :Alyzr(i):Histogram:FsWithDivider?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:FsWithDivider] Value

**Response Argument(s):** Value <int> {hdFswithdiv1=0 | hdFswithdiv2=1 | hdFswithdiv4=2 | hdFswithdiv8=3 | hdFswithdiv16=4 | hdFswithdiv32=5 | hdFswithdiv64=6 | hdFswithdiv128=7 | hdFswithdiv256=8 | hdFswithdiv512=9}

**Example:** :Alyzr(0):Histogram:FsWithDivider?  
[:Alyzr(0):Histogram:FsWithDivider] hdFswithdiv1

**Related Command(s):** FswithDivider

**Description:** Queries the value of the Fswith divider. If the divider is set to "/4", for example, the analyzer only examines every 4th point in the input data stream.

### FswithDivider

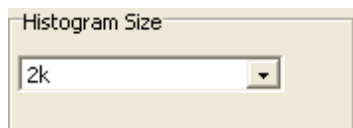
**Command Syntax:** :Alyzr(i):Histogram:FswithDivider Value [, AllowCoercion]

**Command Argument(s):** Value <int> {hdFswithdiv1=0 | hdFswithdiv2=1 | hdFswithdiv4=2 | hdFswithdiv8=3 | hdFswithdiv16=4 | hdFswithdiv32=5 | hdFswithdiv64=6 | hdFswithdiv128=7 | hdFswithdiv256=8 | hdFswithdiv512=9}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:FswithDivider hdFswithdiv1

**Related Command(s):** FswithDivider?

**Description:** Sets the value of the Fswith divider. If the divider is set to "/4", for example, the analyzer only examines every 4th point in the input data stream.



## SampleSize?

**Command Syntax:** :Alyzr(i):Histogram:SampleSize?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:SampleSize] Value

**Response Argument(s):** Value <int> {hs64k=0 | hs32k=1 | hs16k=2 | hs8k=3 | hs4k=4 | hs2k=5 | hs1k=6 | hs512=7}

**Example:** :Alyzr(0):Histogram:SampleSize?

**[ :Alyzr(0):Histogram:SampleSize ] hs64k**

**Related Command(s):** SampleSize

**Description:** Queries the number of input points examined for each histogram.

## SampleSize

**Command Syntax:** :Alyzr(i):Histogram:SampleSize Value [, AllowCoercion]

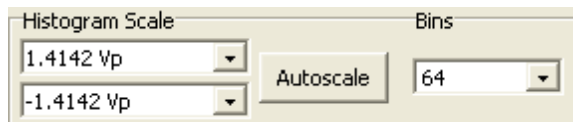
**Command Argument(s):** Value <int> {hs64k=0 | hs32k=1 | hs16k=2 | hs8k=3 | hs4k=4 | hs2k=5 | hs1k=6 | hs512=7}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:SampleSize hs64k

**Related Command(s):** SampleSize?

**Description:** Sets the number of input points examined for each histogram.



## AnlgScaleMax?

**Command Syntax:** :Alyzr(i):Histogram:AnlgScaleMax? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Alyzr(i):Histogram:AnlgScaleMax] Value

**Response Argument(s):** Value <unit>

**Example:** :Alyzr(0):Histogram:AnlgScaleMax?

**[ :Alyzr(0):Histogram:AnlgScaleMax ] Value**

**Related Command(s):** AnlgScaleMax

**Description:** Queries the input value corresponding to the maximum bin (for Analog inputs).

## AnlgScaleMax

**Command Syntax:** :Alyzr(i):Histogram:AnlgScaleMax Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:AnlgScaleMax Value

**Related Command(s):** AnlgScaleMax?

**Description:** Sets the input value corresponding to the maximum bin (for Analog inputs).



## AnlgScaleMin?

**Command Syntax:** :Alyzr(i):Histogram:AnlgScaleMin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Histogram:AnlgScaleMin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Histogram:AnlgScaleMin?  
[:Alyzr(0):Histogram:AnlgScaleMin] Value

**Related Command(s):** AnlgScaleMin

**Description:** Queries the input value corresponding to the minimum bin (for Analog inputs).

## AnlgScaleMin

**Command Syntax:** :Alyzr(i):Histogram:AnlgScaleMin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Histogram:AnlgScaleMin *Value*

**Related Command(s):** AnlgScaleMin?

**Description:** Sets the input value corresponding to the minimum bin (for Analog inputs).

## DigScaleMax?

**Command Syntax:** :Alyzr(i):Histogram:DigScaleMax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Histogram:DigScaleMax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Histogram:DigScaleMax?  
[:Alyzr(0):Histogram:DigScaleMax] Value

**Related Command(s):** DigScaleMax

**Description:** Queries the input value corresponding to the maximum bin (for Digital Audio inputs).

## DigScaleMax

**Command Syntax:** :Alyzr(i):Histogram:DigScaleMax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Alyzr(0):Histogram:DigScaleMax *Value*

**Related Command(s):** DigScaleMax?

**Description:** Sets the input value corresponding to the maximum bin (for Digital Audio inputs).

## DigScaleMin?

**Command Syntax:** :Alyzr(i):Histogram:DigScaleMin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Histogram:DigScaleMin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Histogram:DigScaleMin?  
[:Alyzr(0):Histogram:DigScaleMin] **Value**

**Related Command(s):** DigScaleMin

**Description:** Queries the input value corresponding to the minimum bin (for Digital Audio inputs).

## DigScaleMin

**Command Syntax:** :Alyzr(i):Histogram:DigScaleMin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:DigScaleMin *Value*

**Related Command(s):** DigScaleMin?

**Description:** Sets the input value corresponding to the minimum bin (for Digital Audio inputs).

## AutoScale

**Command Syntax:** :Alyzr(i):Histogram:AutoScale

**Command Argument(s):** None

**Example:** :Alyzr(0):Histogram:AutoScale

**Description:** Sets the Scale Min/Max to the minimum and maximum value of the current input data.

## NumBins?

**Command Syntax:** :Alyzr(i):Histogram:NumBins?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:NumBins] *Value*

**Response Argument(s):** *Value* <int> {hb16=0 | hb32=1 | hb64=2 | hb128=3 | hb256=4 | hb512=5}

**Example:** :Alyzr(0):Histogram:NumBins?  
[:Alyzr(0):Histogram:NumBins] **hb16**

**Related Command(s):** NumBins

**Description:** Queries the number of histogram bins.

## NumBins

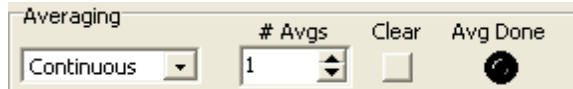
**Command Syntax:** :Alyzr(i):Histogram:NumBins *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {hb16=0 | hb32=1 | hb64=2 | hb128=3 | hb256=4 | hb512=5}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:NumBins hb16

**Related Command(s):** NumBins?

**Description:** Sets the number of histogram bins.



## Averaging?

**Command Syntax:** :Alyzr(i):Histogram:Averaging?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:Averaging] *Value*

**Response Argument(s):** *Value* <int> {haContinuous=0 | haSingle=1}

**Example:** :Alyzr(0):Histogram:Averaging?

**[ :Alyzr(0):Histogram:Averaging] haContinuous**

**Related Command(s):** Averaging

**Description:** Queries the averaging type.

## Averaging

**Command Syntax:** :Alyzr(i):Histogram:Averaging *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {haContinuous=0 | haSingle=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:Averaging haContinuous

**Related Command(s):** Averaging?

**Description:** Sets the averaging type.

## AvgDone?

**Command Syntax:** :Alyzr(i):Histogram:AvgDone?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:AvgDone] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:AvgDone?

**[ :Alyzr(0):Histogram:AvgDone] False**

**Description:** Queries the "average done" status.

## NumAverages?

**Command Syntax:** :Alyzr(i):Histogram:NumAverages?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:NumAverages] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Alyzr(0):Histogram:NumAverages?

**[ :Alyzr(0):Histogram:NumAverages ] 10**

**Related Command(s):** NumAverages

**Description:** Queries the number of averages.

## NumAverages

**Command Syntax:** :Alyzr(i):Histogram:NumAverages *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:NumAverages 10

**Related Command(s):** NumAverages?

**Description:** Sets the number of averages.

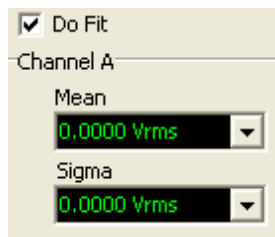
## ResetAvg

**Command Syntax:** :Alyzr(i):Histogram:ResetAvg

**Command Argument(s):** None

**Example:** :Alyzr(0):Histogram:ResetAvg

**Description:** Resets the average buffer.



## DoFit?

**Command Syntax:** :Alyzr(i):Histogram:DoFit?

**Command Argument(s):**

**Response Syntax:** [:Alyzr(i):Histogram:DoFit] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Alyzr(0):Histogram:DoFit?

**[ :Alyzr(0):Histogram:DoFit ] False**

**Related Command(s):** DoFit

**Description:** Queries the on/off status of the exponential fit.

## DoFit

**Command Syntax:** `:Alyzr(i):Histogram:DoFit Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Alyzr(0):Histogram:DoFit False`

**Related Command(s):** DoFit?

**Description:** Sets the on/off status of the exponential fit.

## MeanA?

**Command Syntax:** `:Alyzr(i):Histogram:MeanA? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Histogram:MeanA] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Histogram:MeanA?`  
`[:Alyzr(0):Histogram:MeanA] 0.012 Vp`

**Description:** Queries the average (mean) value of the A channel input data.

## MeanB?

**Command Syntax:** `:Alyzr(i):Histogram:MeanB? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Histogram:MeanB] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Histogram:MeanB?`  
`[:Alyzr(0):Histogram:MeanB] -0.12 Vp`

**Description:** Queries the average (mean) value of the A channel input data.

## StdDevA?

**Command Syntax:** `:Alyzr(i):Histogram:StdDevA? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:Alyzr(i):Histogram:StdDevA] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:Alyzr(0):Histogram:StdDevA?`  
`[:Alyzr(0):Histogram:StdDevA]`

**Description:** Queries the standard deviation of the A channel input data.

## StdDevB?

**Command Syntax:** :Alyzr(i):Histogram:StdDevB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Alyzr(i):Histogram:StdDevB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Alyzr(0):Histogram:StdDevB?  
[:Alyzr(0):Histogram:StdDevB] **Value**

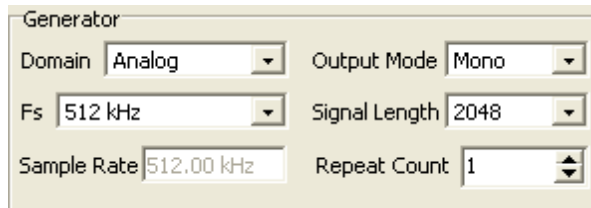
**Description:** Queries the standard deviation of the B channel input data..

## Supported Form Commands:

**:Alyzr(i):HIST:OpenForm**  
**:Alyzr(i):HIST:OpenFormwID?**  
**:Alyzr(i):HIST:CloseForm**  
**:Alyzr(i):HIST:CloseForms**  
**:Alyzr(i):HIST:FormCount?**  
**:Alyzr(i):HIST:FormID?**

### 2.3.8 Multitone Configuration

<b>Object:</b>	<b>:MultiToneCfg</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the configuration of the multitone signal (analog or digital).



#### Generator?

**Command Syntax:** :MultiToneCfg:Generator?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:Generator] Value

**Response Argument(s):** Value <int> {Anlg=0 | Dig=1}

**Example:** :MultiToneCfg:Generator?  
[:MultiToneCfg:Generator] Anlg

**Related Command(s):** Generator

**Description:** Queries the generator selection, analog or digital for multitone analysis.

#### Generator

**Command Syntax:** :MultiToneCfg:Generator Value [, AllowCoercion]

**Command Argument(s):** Value <int> {Anlg=0 | Dig=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:Generator Anlg

**Related Command(s):** Generator?

**Description:** Sets the generator selection, analog or digital for multitone analysis

#### Stereo?

**Command Syntax:** :MultiToneCfg:Stereo?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:Stereo] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :MultiToneCfg:Stereo?  
[:MultiToneCfg:Stereo] False

**Related Command(s):** Stereo

**Description:** Queries the stereo/mono selection for multitone generation and analysis.

## Stereo

**Command Syntax:** `:MultiToneCfg:Stereo Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:MultiToneCfg:Stereo False`

**Related Command(s):** Stereo?

**Description:** Sets the stereo/mono selection for multitone generation and analysis.

## Fs?

**Command Syntax:** `:MultiToneCfg:FsWith?`

**Command Argument(s):**

**Response Syntax:** `[:MultiToneCfg:FsWith] Value`

**Response Argument(s):** `Value <int> {mtHz512k=0 | mtHz128k=1 | mtHz64k=2 | mtISR=3 | mtOSR=4}`

**Example:** `:MultiToneCfg:FsWith?`  
`[:MultiToneCfg:FsWith] mtHz512k`

**Related Command(s):** FsWith

**Description:** Queries the generator sample rate mode for multitone generation.

## FsWith

**Command Syntax:** `:MultiToneCfg:FsWith Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {mtHz512k=0 | mtHz128k=1 | mtHz64k=2 | mtISR=3 | mtOSR=4}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:MultiToneCfg:FsWith mtHz512k`

**Related Command(s):** FsWith?

**Description:** Sets the generator sample rate mode for multitone generation.

## Length?

**Command Syntax:** `:MultiToneCfg:Length?`

**Command Argument(s):**

**Response Syntax:** `[:MultiToneCfg:Length] Value`

**Response Argument(s):** `Value <int> {ml512=0 | ml1k=1 | ml2k=2 | ml4k=3 | ml8k=4 | ml16k=5 | ml32k=6}`

**Example:** `:MultiToneCfg:Length?`  
`[:MultiToneCfg:Length] ml512`

**Related Command(s):** Length

**Description:** Queries the length of the multitone signal in samples.



## Length

**Command Syntax:** `:MultiToneCfg:Length Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {m1512=0 | m1k=1 | m2k=2 | m4k=3 | m8k=4 | m16k=5 | m32k=6}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:MultiToneCfg:Length m1512`

**Related Command(s):** Length?

**Description:** Sets the length of the multitone signal in samples.

## SampleRate?

**Command Syntax:** `:MultiToneCfg:SampleRate? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:MultiToneCfg:SampleRate] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:MultiToneCfg:SampleRate?`  
`[ :MultiToneCfg:SampleRate ] 128000.0 Hz`

**Related Command(s):** SampleRate

**Description:** Queries the actual value of the multitone generator sample rate.

## RepeatCount?

**Command Syntax:** `:MultiToneCfg:RepeatCount?`

**Command Argument(s):**

**Response Syntax:** `[:MultiToneCfg:RepeatCount] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:MultiToneCfg:RepeatCount?`  
`[ :MultiToneCfg:RepeatCount ] 4`

**Related Command(s):** RepeatCount

**Description:** Queries the generator repeat count. The repeat count determines how many repetitions of the multitone signal will occur in each burst.

## RepeatCount

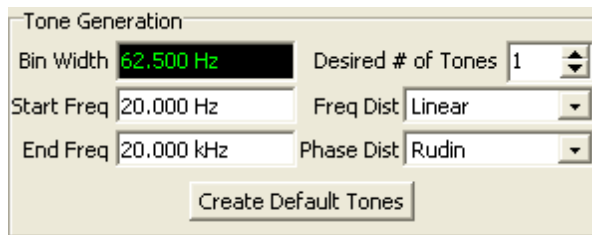
**Command Syntax:** `:MultiToneCfg:RepeatCount Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:MultiToneCfg:RepeatCount 4`

**Related Command(s):** RepeatCount?

**Description:** Sets the generator repeat count. The repeat count determines how many repetitions of the multitone signal will occur in each burst.



## BinWidthRdg?

**Command Syntax:** :MultiToneCfg:BinWidthRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:BinWidthRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:BinWidthRdg?  
[:MultiToneCfg:BinWidthRdg] 62.5 Hz

**Description:** Queries the minimum tone separation determined by the generator sample rate and signal length.

## NumTonesNominal?

**Command Syntax:** :MultiToneCfg:NumTonesNominal?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:NumTonesNominal] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :MultiToneCfg:NumTonesNominal?  
[:MultiToneCfg:NumTonesNominal] Value

**Related Command(s):** NumTonesNominal

**Description:** Queries the desired number of tones.

## NumTonesNominal

**Command Syntax:** :MultiToneCfg:NumTonesNominal *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :MultiToneCfg:NumTonesNominal Value

**Related Command(s):** NumTonesNominal?

**Description:** Sets the desired number of tones.

## StartFreq?

**Command Syntax:** :MultiToneCfg:StartFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:StartFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:StartFreq?  
[:MultiToneCfg:StartFreq] **Value**

**Related Command(s):** StartFreq

**Description:** Queries the minimum tone frequency.

## StartFreq

**Command Syntax:** :MultiToneCfg:StartFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:StartFreq *Value*

**Related Command(s):** StartFreq?

**Description:** Sets the minimum tone frequency.

## StopFreq?

**Command Syntax:** :MultiToneCfg:StopFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:StopFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:StopFreq?  
[:MultiToneCfg:StopFreq] **Value**

**Related Command(s):** StopFreq

**Description:** Queries the maximum tone frequency.

## StopFreq

**Command Syntax:** :MultiToneCfg:StopFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:StopFreq *Value*

**Related Command(s):** StopFreq?

**Description:** Sets the maximum tone frequency.

## FreqDist?

**Command Syntax:** :MultiToneCfg:FreqDist?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:FreqDist] Value

**Response Argument(s):** Value <int> {mfLinear=0 | mfLog=1 | mfOctave=2 | mfDecade=3 | mfPrime=4 | mfLinearPrime=5 | mfLogPrime=6}

**Example:** :MultiToneCfg:FreqDist?  
[:MultiToneCfg:FreqDist] mfLinear

**Related Command(s):** FreqDist

**Description:** Queries the tone distribution algorithm.

## FreqDist

**Command Syntax:** :MultiToneCfg:FreqDist Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mfLinear=0 | mfLog=1 | mfOctave=2 | mfDecade=3 | mfPrime=4 | mfLinearPrime=5 | mfLogPrime=6}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:FreqDist mfLinear

**Related Command(s):** FreqDist?

**Description:** Sets the tone distribution algorithm.

## PhaseDist?

**Command Syntax:** :MultiToneCfg:PhaseDist?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:PhaseDist] Value

**Response Argument(s):** Value <int> {mpZero=0 | mpNewman=1 | mpSchroeder=2 | mpZygmund=3 | mpRudin=4 | mpRandom=5}

**Example:** :MultiToneCfg:PhaseDist?  
[:MultiToneCfg:PhaseDist] mpZero

**Related Command(s):** PhaseDist

**Description:** Queries the phase distribution algorithm.

## PhaseDist

**Command Syntax:** :MultiToneCfg:PhaseDist Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mpZero=0 | mpNewman=1 | mpSchroeder=2 | mpZygmund=3 | mpRudin=4 | mpRandom=5}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:PhaseDist mpZero

**Related Command(s):** PhaseDist?

**Description:** Sets the phase distribution algorithm.

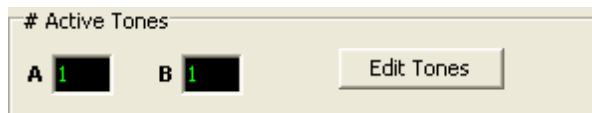
## DefaultTones

**Command Syntax:** :MultiToneCfg:DefaultTones

**Command Argument(s):** None

**Example:** :MultiToneCfg:DefaultTones

**Description:** Calculates a set of tones from the specified frequency limits and frequency and phase distributions.



## NumTonesActualA?

**Command Syntax:** :MultiToneCfg:NumTonesActualA?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:NumTonesActualA] Value

**Response Argument(s):** Value <int>

**Example:** :MultiToneCfg:NumTonesActualA?  
[:MultiToneCfg:NumTonesActualA] Value

**Description:** Queries the actual number of A-channel tones.

## NumTonesActualB?

**Command Syntax:** :MultiToneCfg:NumTonesActualB?

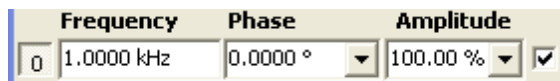
**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:NumTonesActualB] Value

**Response Argument(s):** Value <int>

**Example:** :MultiToneCfg:NumTonesActualB?  
[:MultiToneCfg:NumTonesActualB] Value

**Description:** Queries the actual number of B-channel tones.



## RelToneNumA?

**Command Syntax:** :MultiToneCfg:RelToneNumA?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:RelToneNumA] Value

**Response Argument(s):** Value <int>

**Example:** :MultiToneCfg:RelToneNumA?  
[:MultiToneCfg:RelToneNumA] Value

**Related Command(s):** RelToneNumA

**Description:** Queries the index of the A-channel tone that will be used for tone-relative calculations.

## RelToneNumA

**Command Syntax:** :MultiToneCfg:RelToneNumA Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:RelToneNumA Value

**Related Command(s):** RelToneNumA?

**Description:** Sets the index of the A-channel tone that will be used for tone-relative calculations.

## RelToneNumB?

**Command Syntax:** :MultiToneCfg:RelToneNumB?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:RelToneNumB] Value

**Response Argument(s):** Value <int>

**Example:** :MultiToneCfg:RelToneNumB?  
[:MultiToneCfg:RelToneNumB] Value

**Related Command(s):** RelToneNumB

**Description:** Queries the index of the B-channel tone that will be used for tone-relative calculations.

## RelToneNumB

**Command Syntax:** :MultiToneCfg:RelToneNumB Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:RelToneNumB Value

**Related Command(s):** RelToneNumB?

**Description:** Sets the index of the B-channel tone that will be used for tone-relative calculations.

Sort Tones

Recalc Phase

## OptimizePhase

**Command Syntax:** :MultiToneCfg:OptimizePhase *Channel*

**Command Argument(s):** *Channel* <int> {chanA=0|chanB=1}

**Example:** :MultiToneCfg:OptimizePhasechanA

**Description:** Calculates the tone phases based on the selected phase distribution.

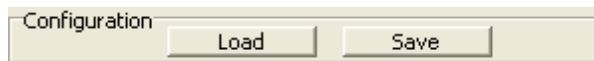
## SortTones

**Command Syntax:** :MultiToneCfg:SortTones *Channel*

**Command Argument(s):** *Channel* <int> {chanA=0|chanB=1}

**Example:** :MultiToneCfg:SortToneschanA

**Description:** Sorts the tones in order of increasing frequency.



## Load

**Command Syntax:** :MultiToneCfg:Load *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :MultiToneCfg:Load Myfile.xml

**Description:** Loads the specified multitone configuration file.

## Save

**Command Syntax:** :MultiToneCfg:Save *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :MultiToneCfg:Save Myfile.xml

**Description:** Saves the current multitone configuration to a file.

## CloseTonesForm

**Command Syntax:** :MultiToneCfg:CloseTonesForm *FormID*

**Command Argument(s):** *FormID* <int>

**Example:** :MultiToneCfg:CloseTonesForm 21

**Description:** Close the tone form with the specified FormID.

## CloseTonesForms

**Command Syntax:** :MultiToneCfg:CloseTonesForms

**Command Argument(s):** None

**Example:** :MultiToneCfg:CloseTonesForms

**Description:** Close all tone forms on all pages of the page control.

## OpenTonesForm

**Command Syntax:** :MultiToneCfg:OpenTonesForm

**Command Argument(s):** None

**Example:** :MultiToneCfg:OpenTonesForm

**Description:** Open a tone form on the current page of the page control.

## OpenTonesFormwID?

**Command Syntax:** :MultiToneCfg:OpenTonesFormwID?

**Command Argument(s):** None

**Response Syntax:** [:MultiToneCfg:OpenTonesFormwID] FormID

**Response Argument(s):** FormID <int>

**Example:** :MultiToneCfg:OpenTonesFormwID?  
[:MultiToneCfg:OpenTonesFormwID] 21

**Description:** Opens a tone form on the current page of the page control and returns the FormID of the newly opened form.

## TonesFormCount?

**Command Syntax:** :MultiToneCfg:TonesFormCount?

**Command Argument(s):** None

**Response Syntax:** [:MultiToneCfg:TonesFormCount] Count

**Response Argument(s):** Count <int>

**Example:** :MultiToneCfg:TonesFormCount?  
[:MultiToneCfg:TonesFormCount] 3

**Description:** Queries the total number of tone forms found on all pages of the page control.

## TonesFormID?

**Command Syntax:** :MultiToneCfg:TonesFormID? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:MultiToneCfg:TonesFormID] FormID

**Response Argument(s):** FormID <int>

**Example:** :MultiToneCfg:TonesFormID? 0  
[:MultiToneCfg:TonesFormID] 21

**Description:** Returns the FormID of the  $i^{\text{th}}$  tone form.



---

*Supported Form Commands:*

**:MultiToneCfg:OpenForm**  
**:MultiToneCfg:OpenFormwID?**  
**:MultiToneCfg:CloseForm**  
**:MultiToneCfg:CloseForms**  
**:MultiToneCfg:FormCount?**  
**:MultiToneCfg:FormID?**

## 2.3.8.1 ToneA

<b>Object:</b>	<b>:MultiToneCfg:ToneA(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0   1   2   3   ...   48   49}
<i>Description:</i>	Commands related to the <i>i</i> <sup>th</sup> A ch Multitone tone.

**Amplitude?**

**Command Syntax:** :MultiToneCfg:ToneA(i):Amplitude? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:ToneA(i):Amplitude] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:ToneA(0):Amplitude?  
[:MultiToneCfg:ToneA(0):Amplitude] 80 pct

**Related Command(s):** Amplitude

**Description:** Queries the amplitude of the tone. Tone amplitudes are specified in relative units.

**Amplitude**

**Command Syntax:** :MultiToneCfg:ToneA(i):Amplitude *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:ToneA(0):Amplitude 80.0

**Related Command(s):** Amplitude?

**Description:** Sets the amplitude of the tone. Tone amplitudes are specified in relative units..

**Freq?**

**Command Syntax:** :MultiToneCfg:ToneA(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:ToneA(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:ToneA(0):Freq?  
[:MultiToneCfg:ToneA(0):Freq] 1000 Hz

**Related Command(s):** Freq

**Description:** Queries the frequency of the tone..

## Freq

**Command Syntax:** `:MultiToneCfg:ToneA(i):Freq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:MultiToneCfg:ToneA(0):Freq 1000.0`

**Related Command(s):** `Freq?`

**Description:** Sets the frequency of the tone.

## On?

**Command Syntax:** `:MultiToneCfg:ToneA(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:MultiToneCfg:ToneA(i):On] Value`

**Response Argument(s):** `Value <int> {False=0|True=1}`

**Example:** `:MultiToneCfg:ToneA(0):On?`  
`[:MultiToneCfg:ToneA(0):On] False`

**Related Command(s):** `On`

**Description:** Queries the on/off status of the tone.

## On

**Command Syntax:** `:MultiToneCfg:ToneA(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0|True=1}`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:MultiToneCfg:ToneA(0):On False`

**Related Command(s):** `On?`

**Description:** Sets the on/off status of the tone.

## Phase?

**Command Syntax:** `:MultiToneCfg:ToneA(i):Phase? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:MultiToneCfg:ToneA(i):Phase] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:MultiToneCfg:ToneA(0):Phase?`  
`[:MultiToneCfg:ToneA(0):Phase] 22 deg`

**Related Command(s):** `Phase`

**Description:** Queries the phase of the tone.

## Phase

**Command Syntax:** :MultiToneCfg:ToneA(i):Phase *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :MultiToneCfg:ToneA(0):Phase 22.0 deg

**Related Command(s):** Phase?

**Description:** Sets the phase of the tone.

## 2.3.8.2 ToneB

<b>Object:</b>	<b>:MultiToneCfg:ToneB(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int> {0   1   2   3   ...   48   49}
<i>Description:</i>	Commands related to the <i>i</i> <sup>th</sup> A ch Multitone tone.

**Amplitude?**

**Command Syntax:** :MultiToneCfg:ToneB(i):Amplitude? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:ToneB(i):Amplitude] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:ToneB(0):Amplitude?  
[:MultiToneCfg:ToneB(0):Amplitude] 80 pct

**Related Command(s):** Amplitude

**Description:** Queries the amplitude of the tone. Tone amplitudes are specified in relative units.

**Amplitude**

**Command Syntax:** :MultiToneCfg:ToneB(i):Amplitude *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:ToneB(0):Amplitude 80 pct

**Related Command(s):** Amplitude?

**Description:** Sets the amplitude of the tone. Tone amplitudes are specified in relative units.

**Freq?**

**Command Syntax:** :MultiToneCfg:ToneB(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:ToneB(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:ToneB(0):Freq?  
[:MultiToneCfg:ToneB(0):Freq] 1000 Hz

**Related Command(s):** Freq

**Description:** Queries the frequency of the tone.

## Freq

**Command Syntax:** :MultiToneCfg:ToneB(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:ToneB(0):Freq *Value*

**Related Command(s):** Freq?

**Description:** Sets the frequency of the tone..

## On?

**Command Syntax:** :MultiToneCfg:ToneB(i):On?

**Command Argument(s):**

**Response Syntax:** [:MultiToneCfg:ToneB(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :MultiToneCfg:ToneB(0):On?  
 [:MultiToneCfg:ToneB(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the tone.

## On

**Command Syntax:** :MultiToneCfg:ToneB(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :MultiToneCfg:ToneB(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the tone.

## Phase?

**Command Syntax:** :MultiToneCfg:ToneB(i):Phase? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:MultiToneCfg:ToneB(i):Phase] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :MultiToneCfg:ToneB(0):Phase?  
 [:MultiToneCfg:ToneB(0):Phase] **22 deg**

**Related Command(s):** Phase

**Description:** Queries the phase of the tone..

## Phase

**Command Syntax:** :MultiToneCfg:ToneB(i):Phase *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :MultiToneCfg:ToneB(0):Phase 22 deg

**Related Command(s):** Phase?

**Description:** Sets the phase of the tone.

### 2.3.9 Analyzer References

<b>Object:</b>	<b>:AlyzrReferences</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the analyzer unit references.

Analog

dBrA 1.0000 Vrms

dBrB 1.0000 Vrms

dBm 600.00 ohms

Watts 8.0000 ohms

Freq 1.0000 kHz

#### AnlgdBrA?

**Command Syntax:** :AlyzrReferences:AnlgdBrA? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:AnlgdBrA] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:AnlgdBrA?  
[:AlyzrReferences:AnlgdBrA] 1 Vrms

**Related Command(s):** AnlgdBrA

**Description:** Queries the analog dBrA reference.

#### AnlgdBrA

**Command Syntax:** :AlyzrReferences:AnlgdBrA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:AnlgdBrA 1 Vrms

**Related Command(s):** AnlgdBrA?

**Description:** Sets the analog dBrA reference.



## AnlgdBrB?

**Command Syntax:** :AlyzrReferences:AnlgdBrB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:AnlgdBrB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:AnlgdBrB?  
[:AlyzrReferences:AnlgdBrB] 1 Vrms

**Related Command(s):** AnlgdBrB

**Description:** Queries the analog dBrB reference.

## AnlgdBrB

**Command Syntax:** :AlyzrReferences:AnlgdBrB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:AnlgdBrB 1 Vrms

**Related Command(s):** AnlgdBrB?

**Description:** Sets the analog dBrB reference.

## AnlgFreq?

**Command Syntax:** :AlyzrReferences:AnlgFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:AnlgFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:AnlgFreq?  
[:AlyzrReferences:AnlgFreq] 1000.0 Hz

**Related Command(s):** AnlgFreq

**Description:** Queries the analog frequency reference.

## AnlgFreq

**Command Syntax:** :AlyzrReferences:AnlgFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:AnlgFreq 1000 Hz

**Related Command(s):** AnlgFreq?

**Description:** Sets the analog frequency reference.

## dBmZ?

**Command Syntax:** :AlyzrReferences:dBmZ? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:dBmZ] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:dBmZ?  
[:AlyzrReferences:dBmZ] 600.0

**Related Command(s):** dBmZ

**Description:** Queries the reference impedance used in dBm calculations.

## dBmZ

**Command Syntax:** :AlyzrReferences:dBmZ *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AlyzrReferences:dBmZ 600.

**Related Command(s):** dBmZ?

**Description:** Sets the reference impedance used in dBm calculations..

## WattsZ?

**Command Syntax:** :AlyzrReferences:WattsZ? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:WattsZ] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:WattsZ?  
[:AlyzrReferences:WattsZ] 8.0

**Related Command(s):** WattsZ

**Description:** Queries the terminating impedance used for calculating Watts.

## WattsZ

**Command Syntax:** :AlyzrReferences:WattsZ *Value* [, *AllowCoercion*]

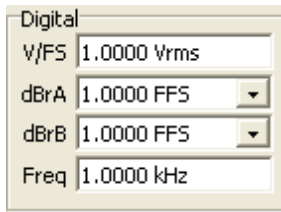
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AlyzrReferences:WattsZ 8.0

**Related Command(s):** WattsZ?

**Description:** Sets the terminating impedance used for calculating Watts.



## DigdBrA?

**Command Syntax:** :AlyzrReferences:DigdBrA? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:DigdBrA] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:DigdBrA?  
[:AlyzrReferences:DigdBrA] 1.0 Vrms

**Related Command(s):** DigdBrA

**Description:** Queries the reference value for digital audio dBrA units.

## DigdBrA

**Command Syntax:** :AlyzrReferences:DigdBrA *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:DigdBrA 1.0 Vrms

**Related Command(s):** DigdBrA?

**Description:** Sets the reference value for digital audio dBrA units.

## DigdBrB?

**Command Syntax:** :AlyzrReferences:DigdBrB? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:DigdBrB] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:DigdBrB?  
[:AlyzrReferences:DigdBrB] 1.0 Vrms

**Related Command(s):** DigdBrB

**Description:** Queries the reference value for digital audio dBrB units.

## DigdBdB

**Command Syntax:** :AlyzrReferences:DigdBdB *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:DigdBdB 1.0

**Related Command(s):** DigdBdB?

**Description:** Sets the reference value for digital audio dBdB units.

## DigFreq?

**Command Syntax:** :AlyzrReferences:DigFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:DigFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:DigFreq?  
 [:AlyzrReferences:DigFreq] 1000.0 Hz

**Related Command(s):** DigFreq

**Description:** Queries the digital audio reference frequency.

## DigFreq

**Command Syntax:** :AlyzrReferences:DigFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AlyzrReferences:DigFreq 1000.0

**Related Command(s):** DigFreq?

**Description:** Sets the digital audio reference frequency

## DigVfs?

**Command Syntax:** :AlyzrReferences:DigVfs? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AlyzrReferences:DigVfs] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AlyzrReferences:DigVfs?  
 [:AlyzrReferences:DigVfs] 1.0 Vrms

**Related Command(s):** DigVfs

**Description:** Queries the digital audio "Volts Full Scale" value.

## DigVfs

**Command Syntax:** :AlyzReferences:DigVfs *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

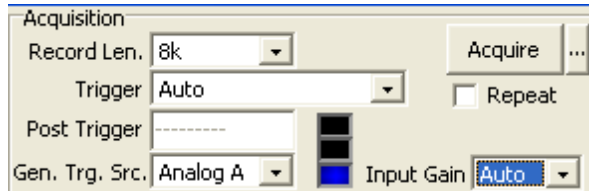
**Example:** :AlyzReferences:DigVfs 1.0

**Related Command(s):** DigVfs?

**Description:** Sets the digital audio "Volts Full Scale" value.

### 2.3.10 Digitizer

<b>Object:</b>	<b>:Digitizer</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the optional digital audio carrier digitizer.



### RecordLength?

**Command Syntax:** :Digitizer:RecordLength?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:RecordLength] Value

**Response Argument(s):** Value <int> {dr14k=0 | dr18k=1 | dr116k=2 | dr132k=3 | dr164k=4 | dr1128k=5 | dr1256k=6 | dr1512k=7 | dr11M=8 | dr12M=9}

**Example:** :Digitizer:RecordLength?

**[ :Digitizer:RecordLength ] dr14k**

**Related Command(s):** RecordLength

**Description:** Queries the digitizer record length.

### RecordLength

**Command Syntax:** :Digitizer:RecordLength Value [, AllowCoercion]

**Command Argument(s):** Value <int> {dr14k=0 | dr18k=1 | dr116k=2 | dr132k=3 | dr164k=4 | dr1128k=5 | dr1256k=6 | dr1512k=7 | dr11M=8 | dr12M=9}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:RecordLength dr14k

**Related Command(s):** RecordLength?

**Description:** Sets the digitizer record length.

## Repeat?

**Command Syntax:** :Digitizer:Repeat?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:Repeat] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Digitizer:Repeat?

**[ :Digitizer:Repeat ] False**

**Related Command(s):** Repeat

**Description:** Queries the digitizer "repeat acquisition" status.

## Repeat

**Command Syntax:** :Digitizer:Repeat *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:Repeat False

**Related Command(s):** Repeat?

**Description:** Sets the digitizer "repeat acquisition" status.

## InputStatusRdg?

**Command Syntax:** :Digitizer:InputStatusRdg?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:InputStatusRdg] *Value*

**Response Argument(s):** *Value* <int> {dgUnderRange=-1 | dgInRange=0 | dgOverRange=1}

**Example:** :Digitizer:InputStatusRdg?

**[ :Digitizer:InputStatusRdg ] dgUnderRange**

**Description:** Queries the digitizer overload status.

## TriggerSource?

**Command Syntax:** :Digitizer:TriggerSource?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:TriggerSource] *Value*

**Response Argument(s):** *Value* <int> {Auto=0 | RcvAC1=1 | RcvBC1=2 | RcvBlkC1=3 | RcvErrC1=4 | RcvAC2=5 | RcvBC2=6 | RcvBlkC2=7 | RcvErrC2=8 | RcvARef=9 | RcvBRef=10 | RcvBlkRef=11 | RcvErrRef=12 | XmtA=13 | XmtB=14 | XmitBlk=15 | RefOutA=16 | RefOutB=17 | RefOutBlk=18 | Generator=19 | ExtRise=20 | ExtFall=21}

**Example:** :Digitizer:TriggerSource?

**[ :Digitizer:TriggerSource ] Auto**

**Related Command(s):** TriggerSource

**Description:** Queries the digitizer trigger source.

## TriggerSource

**Command Syntax:** :Digitizer:TriggerSource *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {Auto=0 | RcvAC1=1 | RcvBC1=2 | RcvBlkC1=3 | RcvErrC1=4 | RcvAC2=5 | RcvBC2=6 | RcvBlkC2=7 | RcvErrC2=8 | RcvARef=9 | RcvBRef=10 | RcvBlkRef=11 | RcvErrRef=12 | XmtA=13 | XmtB=14 | XmitBlk=15 | RefOutA=16 | RefOutB=17 | RefOutBlk=18 | Generator=19 | ExtRise=20 | ExtFall=21}  
**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Digitizer:TriggerSource Auto

**Related Command(s):** TriggerSource?

**Description:** Sets the digitizer trigger source.

## PostTrig?

**Command Syntax:** :Digitizer:PostTrig? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Digitizer:PostTrig] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Digitizer:PostTrig?  
 [:Digitizer:PostTrig] **Value**

**Related Command(s):** PostTrig

**Description:** Queries the percentage of the digitizer record that will follow the trigger.

## PostTrig

**Command Syntax:** :Digitizer:PostTrig *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:PostTrig *Value*

**Related Command(s):** PostTrig?

**Description:** Sets the percentage of the digitizer record that will follow the trigger.

## Acquire

**Command Syntax:** :Digitizer:Acquire *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Digitizer:Acquire ""  
 :Digitizer:Acquire DigData.txt

**Description:** Starts data acquisition. If a filename is given, the data in the file is loaded into the digitizer. If no filename is specified (""), the digitizer acquires data from the current input using the current trigger specifications.



## Cancel

**Command Syntax:** :Digitizer:Cancel

**Command Argument(s):** None

**Example:** :Digitizer:Cancel

**Description:** Cancels any digitizer acquisition in progress.

## Save

**Command Syntax:** :Digitizer:Save *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Digitizer:Save DigFile.txt

**Description:** Saves the current digitizer data to the name file.

## State?

**Command Syntax:** :Digitizer:State?

**Command Argument(s):** None

**Response Syntax:** [:Digitizer:State] *State*

**Response Argument(s):** *State* <int> {dsIdle=0 | dsWaitingForTrig=1 | dsAcquiring=2 | dsAnalyzing=3 | dsCancelling=4}

**Example:** :Digitizer:State?

**[:Digitizer:State] dsIdle**

**Description:** Returns the current state of the digitizer.

## InputGain?

**Command Syntax:** :Digitizer:InputGain?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:InputGain] *Value*

**Response Argument(s):** *Value* <int> {dgAuto=0 | dgx1=1 | dgx2=2 | dgx4=3}

**Example:** :Digitizer:InputGain?

**[:Digitizer:InputGain] dgAuto**

**Related Command(s):** InputGain

**Description:** Queries the digitizer input gain setting.

## InputGain

**Command Syntax:** :Digitizer:InputGain *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {dgAuto=0 | dgx1=1 | dgx2=2 | dgx4=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:InputGain dgAuto

**Related Command(s):** InputGain?

**Description:** Sets the digitizer input gain.



## JitterDetection?

**Command Syntax:** :Digitizer:JitterDetection?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:JitterDetection] Value

**Response Argument(s):** Value <int> {AllBits=0 | StableBits=1 | Preamble=2 | SqRising=3 | SqFalling=4 | SqBoth=5}

**Example:** :Digitizer:JitterDetection?

**[:Digitizer:JitterDetection] AllBits**

**Related Command(s):** JitterDetection

**Description:** Queries the digitizer stable bits selection.

## JitterDetection

**Command Syntax:** :Digitizer:JitterDetection Value [, AllowCoercion]

**Command Argument(s):** Value <int> {AllBits=0 | StableBits=1 | Preamble=2 | SqRising=3 | SqFalling=4 | SqBoth=5}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:JitterDetection AllBits

**Related Command(s):** JitterDetection?

**Description:** Sets the digitizer stable bits selection.

## Reanalyze

**Command Syntax:** :Digitizer:Reanalyze

**Command Argument(s):** None

**Example:** :Digitizer:Reanalyze

**Description:** Reanalyzes the current digitizer data without acquiring new data.

## JitterRecLenRdg?

**Command Syntax:** :Digitizer:JitterRecLenRdg?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:JitterRecLenRdg] Value

**Response Argument(s):** Value <int>

**Example:** :Digitizer:JitterRecLenRdg?  
[:Digitizer:JitterRecLenRdg] 78

**Description:** Queries the total number of jitter points calculated given the digitizer record length and the stable bits selection.

## ClockFreqRdg?

**Command Syntax:** :Digitizer:ClockFreqRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Digitizer:ClockFreqRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Digitizer:ClockFreqRdg?  
[:Digitizer:ClockFreqRdg] 48000 HZ

**Description:** Queries the calculated effective digital audio sampling clock calculated from the digitizer data.

## RMSJitterRdg?

**Command Syntax:** :Digitizer:RMSJitterRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Digitizer:RMSJitterRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Digitizer:RMSJitterRdg?  
[:Digitizer:RMSJitterRdg] 1.3328E-10 SEC

**Description:** Calculates the total jitter calculated from the digitizer data.



## FastInterpolation?

**Command Syntax:** :Digitizer:FastInterpolation?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:FastInterpolation] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Digitizer:FastInterpolation?

**[[:Digitizer:FastInterpolation] False**

**Related Command(s):** FastInterpolation

**Description:** Queries the on/off status of digitizer Fast Interpolation. Turning on fast interpolation enables a faster but less accurated jitter calculation algorithm.

## FastInterpolation

**Command Syntax:** :Digitizer:FastInterpolation *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:FastInterpolation False

**Related Command(s):** FastInterpolation?

**Description:** Sets the on/off status of digitizer Fast Interpolation. Turning on fast interpolation enables a faster but less accurated jitter calculation algorithm.

## RiseFallEdges?

**Command Syntax:** :Digitizer:RiseFallEdges?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:RiseFallEdges] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Digitizer:RiseFallEdges?

**[[:Digitizer:RiseFallEdges] False**

**Related Command(s):** RiseFallEdges

**Description:** Queries the on/off state of the asymmetry correction feature of the digitizer.

## RiseFallEdges

**Command Syntax:** :Digitizer:RiseFallEdges *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:RiseFallEdges False

**Related Command(s):** RiseFallEdges?

**Description:** Sets the on/off state of the asymmetry correction feature of the digitizer.

## DisplayInterp?

**Command Syntax:** :Digitizer:DisplayInterp?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:DisplayInterp] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Digitizer:DisplayInterp?  
[:Digitizer:DisplayInterp] False

**Related Command(s):** DisplayInterp

**Description:** Queries the on/off status of digitizer display interpolation.

## DisplayInterp

**Command Syntax:** :Digitizer:DisplayInterp Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:DisplayInterp False

**Related Command(s):** DisplayInterp?

**Description:** Sets the on/off status of digitizer display interpolation.

## AsymmetryRdg?

**Command Syntax:** :Digitizer:AsymmetryRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Digitizer:AsymmetryRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Digitizer:AsymmetryRdg?  
[:Digitizer:AsymmetryRdg] -0.170039

**Description:** Queries the assymetry value calculated by the digitizer.

## JitterFsRdg?

**Command Syntax:** :Digitizer:JitterFsRdg? [ValueUnit]

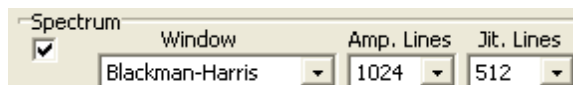
**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Digitizer:JitterFsRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :Digitizer:JitterFsRdg?  
[:Digitizer:JitterFsRdg] 768000

**Description:** Queries the effective jitter sampling rate determined by the input digital audio sampling rate and the jitter stable bits selection.



## CalcSpectrum?

**Command Syntax:** :Digitizer:CalcSpectrum?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:CalcSpectrum] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Digitizer:CalcSpectrum?  
[:Digitizer:CalcSpectrum] False

**Related Command(s):** CalcSpectrum

**Description:** Queries the on/off status of digitizer spectrum measurement calculation.

## CalcSpectrum

**Command Syntax:** :Digitizer:CalcSpectrum Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:CalcSpectrum False

**Related Command(s):** CalcSpectrum?

**Description:** Sets the on/off status of digitizer spectrum measurement calculation.

## SpectrumWindow?

**Command Syntax:** :Digitizer:SpectrumWindow?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:SpectrumWindow] Value

**Response Argument(s):** Value <int> {BlkmanHarris=0 | Hann=1 | Flattop=2 | EqRipple=3 | Uniform=4}

**Example:** :Digitizer:SpectrumWindow?  
[:Digitizer:SpectrumWindow] BlkmanHarris

**Related Command(s):** SpectrumWindow

**Description:** Queries the window selection for digitizer spectrum measurements.

## SpectrumWindow

**Command Syntax:** :Digitizer:SpectrumWindow Value [, AllowCoercion]

**Command Argument(s):** Value <int> {BlkmanHarris=0 | Hann=1 | Flattop=2 | EqRipple=3 | Uniform=4}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:SpectrumWindow BlkmanHarris

**Related Command(s):** SpectrumWindow?

**Description:** Sets the window selection for digitizer spectrum measurements.

## InputSpecLines?

**Command Syntax:** :Digitizer:InputSpecLines?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:InputSpecLines] Value

**Response Argument(s):** Value <int> {il512=0 | il1k=1 | il2k=2 | il4k=3 | il8k=4 | il16k=5}

**Example:** :Digitizer:InputSpecLines?  
[:Digitizer:InputSpecLines] i1512

**Related Command(s):** InputSpecLines

**Description:** Queries the number of lines of resolution for the input amplitude spectra.

## InputSpecLines

**Command Syntax:** :Digitizer:InputSpecLines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {il512=0 | il1k=1 | il2k=2 | il4k=3 | il8k=4 | il16k=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:InputSpecLines i1512

**Related Command(s):** InputSpecLines?

**Description:** Sets the number of lines of resolution for the input amplitude spectra.

## JitterSpecLines?

**Command Syntax:** :Digitizer:JitterSpecLines?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:JitterSpecLines] Value

**Response Argument(s):** Value <int> {j1256=0 | j1512=1 | j11k=2 | j12k=3 | j14k=4 | j18k=5}

**Example:** :Digitizer:JitterSpecLines?  
[:Digitizer:JitterSpecLines] j1256

**Related Command(s):** JitterSpecLines

**Description:** Queries the number of lines of resolution for the jitter spectra

## JitterSpecLines

**Command Syntax:** :Digitizer:JitterSpecLines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {j1256=0 | j1512=1 | j11k=2 | j12k=3 | j14k=4 | j18k=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:JitterSpecLines j1256

**Related Command(s):** JitterSpecLines?

**Description:** Sets the number of lines of resolution for the jitter spectra.



## CalcProbability?

**Command Syntax:** :Digitizer:CalcProbability?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:CalcProbability] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Digitizer:CalcProbability?  
[:Digitizer:CalcProbability] False

**Related Command(s):** CalcProbability

**Description:** Queries the on/off status of digitizer probability measurement calculation..

## CalcProbability

**Command Syntax:** :Digitizer:CalcProbability Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:CalcProbability False

**Related Command(s):** CalcProbability?

**Description:** Sets the on/off status of digitizer probability measurement calculation.

## NumBins?

**Command Syntax:** :Digitizer:NumBins?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:NumBins] Value

**Response Argument(s):** Value <int> {eb64=0 | eb128=1 | eb256=2 | eb512=3 | eb1024=4}

**Example:** :Digitizer:NumBins?  
[:Digitizer:NumBins] eb64

**Related Command(s):** NumBins

**Description:** Queries the number of histogram bins for digitizer probability measurements.

## NumBins

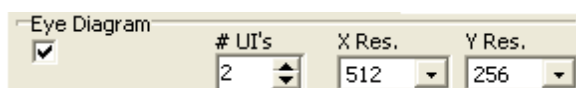
**Command Syntax:** :Digitizer:NumBins Value [, AllowCoercion]

**Command Argument(s):** Value <int> {eb64=0 | eb128=1 | eb256=2 | eb512=3 | eb1024=4}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Digitizer:NumBins eb64

**Related Command(s):** NumBins?

**Description:** Sets the number of histogram bins for digitizer probability measurements





## CalcEyeDiag?

**Command Syntax:** :Digitizer:CalcEyeDiag?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:CalcEyeDiag] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Digitizer:CalcEyeDiag?  
[:Digitizer:CalcEyeDiag] False

**Related Command(s):** CalcEyeDiag

**Description:** Queries the on/off status of eye-diagram calculations.

## CalcEyeDiag

**Command Syntax:** :Digitizer:CalcEyeDiag *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:CalcEyeDiag False

**Related Command(s):** CalcEyeDiag?

**Description:** Sets the on/off status of eye-diagram calculations.

## NumEyes?

**Command Syntax:** :Digitizer:NumEyes?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:NumEyes] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Digitizer:NumEyes?  
[:Digitizer:NumEyes] 2

**Related Command(s):** NumEyes

**Description:** Queries the number of eyes (unit intervals) in the eye-diagram.

## NumEyes

**Command Syntax:** :Digitizer:NumEyes *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Digitizer:NumEyes 2

**Related Command(s):** NumEyes?

**Description:** Sets the number of eyes (unit intervals) in the eye-diagram.

## EyeXRez?

**Command Syntax:** :Digitizer:EyeXRez?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:EyeXRez] Value

**Response Argument(s):** Value <int> {er128=0 | er256=1 | er512=2 | er1024=3}

**Example:** :Digitizer:EyeXRez?

**[ :Digitizer:EyeXRez ] er128**

**Related Command(s):** EyeXRez

**Description:** Quereis the X-resolution of the eye-diagram.

## EyeXRez

**Command Syntax:** :Digitizer:EyeXRez Value [, AllowCoercion]

**Command Argument(s):** Value <int> {er128=0 | er256=1 | er512=2 | er1024=3}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Digitizer:EyeXRez er128

**Related Command(s):** EyeXRez?

**Description:** Sets the X-resolution of the eye-diagram.

## EyeYRez?

**Command Syntax:** :Digitizer:EyeYRez?

**Command Argument(s):**

**Response Syntax:** [:Digitizer:EyeYRez] Value

**Response Argument(s):** Value <int> {er128=0 | er256=1 | er512=2 | er1024=3}

**Example:** :Digitizer:EyeYRez?

**[ :Digitizer:EyeYRez ] er128**

**Related Command(s):** EyeYRez

**Description:** Queries the Y-resolution of the eye-diagram.

## EyeYRez

**Command Syntax:** :Digitizer:EyeYRez Value [, AllowCoercion]

**Command Argument(s):** Value <int> {er128=0 | er256=1 | er512=2 | er1024=3}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Digitizer:EyeYRez er128

**Related Command(s):** EyeYRez?

**Description:** Sets the Y-Resolution of the eye-diagram.

## GetEyeRez?

**Command Syntax:** :Digitizer:GetEyeRez?

**Command Argument(s):** None

**Response Syntax:** [:Digitizer:GetEyeRez] NumXY

**Response Argument(s):** NumXY <doublearray>

**Example:** :Digitizer:GetEyeRez?  
[:Digitizer:GetEyeRez] 128,512

**Description:** Returns the number of cells in a single row of the eye diagram (X-Resolution) followed by the number of cells in a single column of the eyediagram (Y-resolution).

## GetXData?

**Command Syntax:** :Digitizer:GetXData? DgtzrMeas, UnitString

**Command Argument(s):** DgtzrMeas <int> {dmInputTimeRec=0 | dmJitterTimeRec=1 | dmInputSpectrum=2 | dmJitterSpectrum=3 | dmInputProb=4 | dmJitterProb=5 | dmWidthProb=6 | dmRateProb=7 | dmEyeIntensity=8 | dmEyeOutUp=9 | dmEyeInUp=10 | dmEyeInLo=11 | dmEyeOutLo=12}  
UnitString <string>

**Response Syntax:** [:Digitizer:GetXData] Data

**Response Argument(s):** Data <doublearray>

**Example:** :Digitizer:GetXData? dmInputTimeRec, "s"  
[:Digitizer:GetXData] "-2.00002122418268E-7,-1.875019897

**Description:** Queries the X-axis data array for the specified measurement.

## GetYData?

**Command Syntax:** :Digitizer:GetYData? DgtzrMeas, UnitString

**Command Argument(s):** DgtzrMeas <int> {dmInputTimeRec=0 | dmJitterTimeRec=1 | dmInputSpectrum=2 | dmJitterSpectrum=3 | dmInputProb=4 | dmJitterProb=5 | dmWidthProb=6 | dmRateProb=7 | dmEyeIntensity=8 | dmEyeOutUp=9 | dmEyeInUp=10 | dmEyeInLo=11 | dmEyeOutLo=12}  
UnitString <string>

**Response Syntax:** [:Digitizer:GetYData] Data

**Response Argument(s):** Data <doublearray>

**Example:** :Digitizer:GetYData? dmInputTimeRec, "Vp"  
[:Digitizer:GetYData] 1.00440907478333,1.01509428024292,1

**Description:** Queries the Y-axis data array for the specified measurement.

## GetZData?

**Command Syntax:** :Digitizer:GetZData? DgtzrMeas, UnitString

**Command Argument(s):** DgtzrMeas <int> {dmInputTimeRec=0 | dmJitterTimeRec=1 | dmInputSpectrum=2 | dmJitterSpectrum=3 | dmInputProb=4 | dmJitterProb=5 | dmWidthProb=6 | dmRateProb=7 | dmEyeIntensity=8 | dmEyeOutUp=9 | dmEyeInUp=10 | dmEyeInLo=11 | dmEyeOutLo=12}  
UnitString <string>

**Response Syntax:** [:Digitizer:GetZData] Data

**Response Argument(s):** Data <doublearray>

**Example:** :Digitizer:GetZData? dmEyeIntensity, ""  
[:Digitizer:GetZData] 0,0,0,0,0,0,0,0,0,0,0,0,.....

**Related Command(s):** :Digitizer:GetEyeRez?

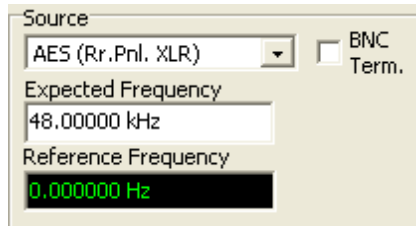
**Description:** Queries the eye diagram intensity data. (Currently, the only supported measurement argument for this command is "8" (dmEyeIntensity)) The intensity data is returned in a "raster-scan" fashion starting from the top left of the eye diagram and ending with the bottom right. The number of cells in the eyediagram can be determined using the GetEyeRez? command.

## Supported Form Commands:

**:Digitizer:OpenForm**  
**:Digitizer:OpenFormwID?**  
**:Digitizer:CloseForm**  
**:Digitizer:CloseForms**  
**:Digitizer:FormCount?**  
**:Digitizer:FormID?**

### 2.3.11 Clock References

<b>Object:</b>	<b>:ClockRef</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Clock Reference panel.



## Source?

**Command Syntax:** :ClockRef:Source?

**Command Argument(s):**

**Response Syntax:** [:ClockRef:Source] Value

**Response Argument(s):** Value <int> {crAES=0 | crExt=1 | crInt=2 | crNTSC=3 | crSECAM=4 | crPAL=5}

**Example:** :ClockRef:Source?

**[ :ClockRef:Source ] crAES**

**Related Command(s):** Source

**Description:** Returns the clock source.

## Source

**Command Syntax:** :ClockRef:Source Value [, AllowCoercion]

**Command Argument(s):** Value <int> {crAES=0 | crExt=1 | crInt=2 | crNTSC=3 | crSECAM=4 | crPAL=5}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :ClockRef:Source crAES

**Related Command(s):** Source?

**Description:** Sets the clock source.

## ExtTerm?

**Command Syntax:** :ClockRef:ExtTerm?

**Command Argument(s):**

**Response Syntax:** [:ClockRef:ExtTerm] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :ClockRef:ExtTerm?

**[ :ClockRef:ExtTerm ] False**

**Related Command(s):** ExtTerm

**Description:** Queries the termination status of the rear-panel BNC clock input.

## ExtTerm

**Command Syntax:** :ClockRef:ExtTerm Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :ClockRef:ExtTerm False

**Related Command(s):** ExtTerm?

**Description:** Sets the termination status of the rear-panel BNC clock input.

## AESFreq?

**Command Syntax:** :ClockRef:AESFreq? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:ClockRef:AESFreq] Value

**Response Argument(s):** Value <unit>

**Example:** :ClockRef:AESFreq?

**[ :ClockRef:AESFreq ] 48000.0 HZ**

**Related Command(s):** AESFreq

**Description:** Queries the expected frequency of the AES reference input.

## AESFreq

**Command Syntax:** :ClockRef:AESFreq Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :ClockRef:AESFreq Value

**Related Command(s):** AESFreq?

**Description:** Sets the expected frequency of the AES reference input..

## ExtFreq?

**Command Syntax:** :ClockRef:ExtFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:ExtFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:ExtFreq?  
[:ClockRef:ExtFreq] 10000000 HZ

**Related Command(s):** ExtFreq

**Description:** Queries the expected frequency for the rear panel external reference BNC input.

## ExtFreq

**Command Syntax:** :ClockRef:ExtFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:ExtFreq 10000000

**Related Command(s):** ExtFreq?

**Description:** Sets the expected frequency for the rear panel external reference BNC input.

## NTSCFreq?

**Command Syntax:** :ClockRef:NTSCFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:NTSCFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:NTSCFreq?  
[:ClockRef:NTSCFreq] 15734.3 HZ

**Related Command(s):** NTSCFreq

**Description:** Queries the expected frequency for the rear panel video input when an NTSC video source is used.

## NTSCFreq

**Command Syntax:** :ClockRef:NTSCFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:NTSCFreq 15734.3 Hz

**Related Command(s):** NTSCFreq?

**Description:** Sets the expected frequency for the rear panel video input when a NTSC video source is used..

## PALFreq?

**Command Syntax:** :ClockRef:PALFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:PALFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:PALFreq?  
[:ClockRef:PALFreq] 15625.0 HZ

**Related Command(s):** PALFreq

**Description:** Queries the expected frequency for the rear panel video input when a PAL video source is used.

## PALFreq

**Command Syntax:** :ClockRef:PALFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:PALFreq 15625.0 HZ

**Related Command(s):** PALFreq?

**Description:** Sets the expected frequency for the rear panel video input when a PAL video source is used..

## SECAMFreq?

**Command Syntax:** :ClockRef:SECAMFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:SECAMFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:SECAMFreq?  
[:ClockRef:SECAMFreq] 15625.0 HZ

**Related Command(s):** SECAMFreq

**Description:** Queries the expected frequency for the rear panel video input when a SECAM video source is used.

## SECAMFreq

**Command Syntax:** :ClockRef:SECAMFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:SECAMFreq 15625.0 Hz

**Related Command(s):** SECAMFreq?

**Description:** Sets the expected frequency for the rear panel video input when a SECAM video source is used.



## FreqRdg?

**Command Syntax:** :ClockRef:FreqRdg? [*ValueUnit*]

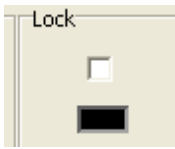
**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:FreqRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:FreqRdg?  
[:ClockRef:FreqRdg] 21896.9 HZ

**Description:** Queries of the frequency of SR1's internal oscillator. When locked, this will match the frequency of the external locking source.



## DoLock?

**Command Syntax:** :ClockRef:DoLock?

**Command Argument(s):**

**Response Syntax:** [:ClockRef:DoLock] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :ClockRef:DoLock?  
[:ClockRef:DoLock] False

**Related Command(s):** DoLock

**Description:** Queries the on/off status for external clock locking. When DoLock? returns true, SR1 will attempt to lock to the specified external clock source. When false, SR1 will run using its own internal clock.

## DoLock

**Command Syntax:** :ClockRef:DoLock *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:DoLock False

**Related Command(s):** DoLock?

**Description:** Sets the on/off status for external clock locking.

## LockStatus?

**Command Syntax:** :ClockRef:LockStatus?

**Command Argument(s):**

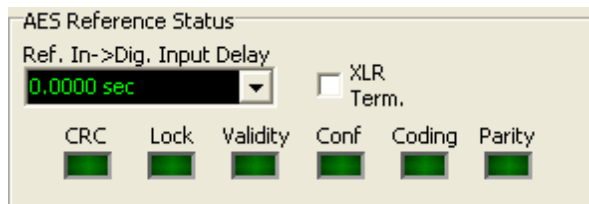
**Response Syntax:** [:ClockRef:LockStatus] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :ClockRef:LockStatus?  
[:ClockRef:LockStatus] False

**Related Command(s):** LockStatus

**Description:** Returns true if SR1 is currently locked to an external source. Returns false if SR1 cannot lock to the specified source.



## RefInDelayRdg?

**Command Syntax:** :ClockRef:RefInDelayRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:ClockRef:RefInDelayRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :ClockRef:RefInDelayRdg?  
[:ClockRef:RefInDelayRdg] 0 S

**Description:** Returns the delay between the rear panel AES reference input and the front-panel digital audio input.

## AESTerm?

**Command Syntax:** :ClockRef:AESTerm?

**Command Argument(s):**

**Response Syntax:** [:ClockRef:AESTerm] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :ClockRef:AESTerm?  
[:ClockRef:AESTerm] False

**Related Command(s):** AESTerm

**Description:** Queries the termination on/off status for the rear-panel AES reference input.

## AESTerm

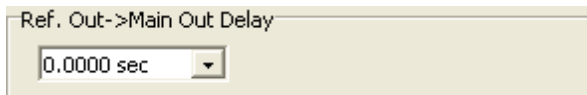
**Command Syntax:** :ClockRef:AESTerm *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:AESTerm False

**Related Command(s):** AESTerm?

**Description:** Sets the termination on/off status for the rear-panel AES reference input.



## DelayOutFromRef?

**Command Syntax:** :ClockRef:DelayOutFromRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:ClockRef:DelayOutFromRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :ClockRef:DelayOutFromRef?  
 [:ClockRef:DelayOutFromRef] 127.500 UI

**Related Command(s):** DelayOutFromRef

**Description:** Queries the delay setting between the AES reference output and the main digital audio. The delay is adjustable between 0 and 128 UI (one full digital audio frame).

## DelayOutFromRef

**Command Syntax:** :ClockRef:DelayOutFromRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :ClockRef:DelayOutFromRef 12.5 UI

**Related Command(s):** DelayOutFromRef?

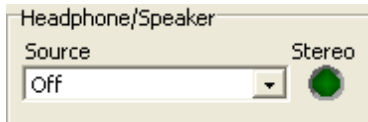
**Description:** Sets the delay between the AES reference output and the main digital audio. The delay is adjustable between 0 and 128 UI (one full digital audio frame).

### *Supported Form Commands:*

**:ClockRef:OpenForm**  
**:ClockRef:OpenFormwID?**  
**:ClockRef:CloseForm**  
**:ClockRef:CloseForms**  
**:ClockRef:FormCount?**  
**:ClockRef:FormID?**

### 2.3.12 Monitors

<b>Object:</b>	<b>:Monitor</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Monitors panel.



### Source?

**Command Syntax:** :Monitor:Source?

**Command Argument(s):**

**Response Syntax:** [:Monitor:Source] Value

**Response Argument(s):** Value <int> {ssOff=0 | ssAnlgGenA=1 | ssAnlgGenB=2 | ssAnlgGenAB=3 | ssAnlgLevelA=4 | ssAnlgLevelB=5 | ssAnlgLevelAB=6 | ssDigLevelA=7 | ssDigLevelB=8 | ssDigLevelAB=9 | ssA0Monitor=10 | ssA1Monitor=11 | ssA0A1Monitor=12 | ssJitter=13}

**Example:** :Monitor:Source?  
[:Monitor:Source] ssOff

**Related Command(s):** Source

**Description:** Queries the source for the Speaker/Headphones.

### Source

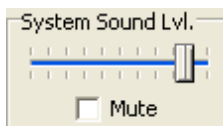
**Command Syntax:** :Monitor:Source Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ssOff=0 | ssAnlgGenA=1 | ssAnlgGenB=2 | ssAnlgGenAB=3 | ssAnlgLevelA=4 | ssAnlgLevelB=5 | ssAnlgLevelAB=6 | ssDigLevelA=7 | ssDigLevelB=8 | ssDigLevelAB=9 | ssA0Monitor=10 | ssA1Monitor=11 | ssA0A1Monitor=12 | ssJitter=13}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:Source ssOff

**Related Command(s):** Source?

**Description:** Sets the source for the Speaker/Headphones.



## SysMute?

**Command Syntax:** :Monitor:SysMute?

**Command Argument(s):**

**Response Syntax:** [:Monitor:SysMute] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Monitor:SysMute?  
[:Monitor:SysMute] False

**Related Command(s):** SysMute

**Description:** Queries the mute status for Windows generated sounds.

## SysMute

**Command Syntax:** :Monitor:SysMute Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:SysMute False

**Related Command(s):** SysMute?

**Description:** Sets the mute status for Windows generated sounds.

## SysSound?

**Command Syntax:** :Monitor:SysSound?

**Command Argument(s):**

**Response Syntax:** [:Monitor:SysSound] Value

**Response Argument(s):** Value <int>

**Example:** :Monitor:SysSound?  
[:Monitor:SysSound] Value

**Related Command(s):** SysSound

**Description:** Queries the Windows sound level.

## SysSound

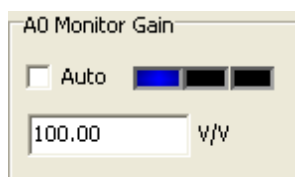
**Command Syntax:** :Monitor:SysSound Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:SysSound Value

**Related Command(s):** SysSound?

**Description:** Sets the Windows sound level.



## A0AutoGain?

**Command Syntax:** :Monitor:A0AutoGain?

**Command Argument(s):**

**Response Syntax:** [:Monitor:A0AutoGain] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Monitor:A0AutoGain?  
[:Monitor:A0AutoGain] False

**Related Command(s):** A0AutoGain

**Description:** Queries the on/off status of A0 monitor autoranging.

## A0AutoGain

**Command Syntax:** :Monitor:A0AutoGain Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:A0AutoGain False

**Related Command(s):** A0AutoGain?

**Description:** Sets the on/off status of A0 monitor autoranging.

## A0Gain?

**Command Syntax:** :Monitor:A0Gain? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Monitor:A0Gain] Value

**Response Argument(s):** Value <unit>

**Example:** :Monitor:A0Gain?  
[:Monitor:A0Gain] 1.0000

**Related Command(s):** A0Gain

**Description:** Queries the A0 monitor gain.

## A0Gain

**Command Syntax:** :Monitor:A0Gain Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:A0Gain 1.0

**Related Command(s):** A0Gain?

**Description:** Sets the A0 monitor gain.

## A0Status?

**Command Syntax:** :Monitor:A0Status?

**Command Argument(s):**

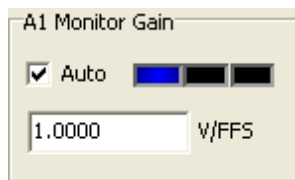
**Response Syntax:** [:Monitor:A0Status] Value

**Response Argument(s):** Value <int> {msUnderRange=0 | msInRange=1 | msOverRange=2}

**Example:** :Monitor:A0Status?

**[:Monitor:A0Status] msUnderRange**

**Description:** Queries the overload status for the A0 monitor.



## A1AutoGain?

**Command Syntax:** :Monitor:A1AutoGain?

**Command Argument(s):**

**Response Syntax:** [:Monitor:A1AutoGain] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Monitor:A1AutoGain?

**[:Monitor:A1AutoGain] False**

**Related Command(s):** A1AutoGain

**Description:** Queries the on/off status of A1 monitor autoranging.

## A1AutoGain

**Command Syntax:** :Monitor:A1AutoGain Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Monitor:A1AutoGain False

**Related Command(s):** A1AutoGain?

**Description:** Sets the on/off status of A1 monitor autoranging.



## A1Gain?

*Command Syntax:* :Monitor:A1Gain? [*ValueUnit*]

*Command Argument(s):* *ValueUnit* <unitstring>

*Response Syntax:* [:Monitor:A1Gain] *Value*

*Response Argument(s):* *Value* <unit>

*Example:* :Monitor:A1Gain?  
[:Monitor:A1Gain] 10.0

*Related Command(s):* A1Gain

*Description:* Queries the A1 Monitor gain.

## A1Gain

*Command Syntax:* :Monitor:A1Gain *Value* [, *AllowCoercion*]

*Command Argument(s):* *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

*Example:* :Monitor:A1Gain 10.0

*Related Command(s):* A1Gain?

*Description:* Sets the A1 Monitor gain.

## A1Status?

*Command Syntax:* :Monitor:A1Status?

*Command Argument(s):*

*Response Syntax:* [:Monitor:A1Status] *Value*

*Response Argument(s):* *Value* <int> {msUnderRange=0 | msInRange=1 | msOverRange=2}

*Example:* :Monitor:A1Status?  
[:Monitor:A1Status] msUnderRange

*Description:* Queries the A1 monitor overload status.

## [Supported Form Commands:](#)

**:Monitor:OpenForm**  
**:Monitor:OpenFormwID?**  
**:Monitor:CloseForm**  
**:Monitor:CloseForms**  
**:Monitor:FormCount?**  
**:Monitor:FormID?**

### 2.3.13 Instrument

<b>Object:</b>	<b>:Instrument</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the instrument user interface.

## Form and Page Control Commands

### ActivePage?

**Command Syntax:** :Instrument:ActivePage?

**Command Argument(s):**

**Response Syntax:** [:Instrument:ActivePage] Value

**Response Argument(s):** Value <int>

**Example:** :Instrument:ActivePage?  
[:Instrument:ActivePage] 2

**Related Command(s):** ActivePage

**Description:** Queries the active page (1-7) of the main page control.

### ActivePage

**Command Syntax:** :Instrument:ActivePage Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Instrument:ActivePage Value

**Related Command(s):** ActivePage?

**Description:** Sets the active page (1-7) of the main page control.

### CloseForm

**Command Syntax:** :Instrument:CloseForm FormID

**Command Argument(s):** FormID <int>

**Example:** :Instrument:CloseForm Value

**Description:** Closes a form of any type, on any page of the page control, with the specified FormID.

## FormCount?

**Command Syntax:** :Instrument:FormCount?

**Command Argument(s):** None

**Response Syntax:** [:Instrument:FormCount] Count

**Response Argument(s):** Count <int>

**Example:** :Instrument:FormCount?  
[:Instrument:FormCount] 2

**Description:** Returns the number of forms on the current page of the page control.

## FormID?

**Command Syntax:** :Instrument:FormID? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:Instrument:FormID] FormID

**Response Argument(s):** FormID <int>

**Example:** :Instrument:FormID? Value  
[:Instrument:FormID] Value

**Description:** Returns the FormID of the Index<sup>th</sup> form on the current page of the page control.

## GetFormPage?

**Command Syntax:** :Instrument:GetFormPage? FormID

**Command Argument(s):** FormID <int>

**Response Syntax:** [:Instrument:GetFormPage] Page

**Response Argument(s):** Page <int>

**Example:** :Instrument:GetFormPage? 101  
[:Instrument:GetFormPage] 2

**Description:** Returns the page of the page control (1-7) containing the form with the given page control.

## GetFormPos?

**Command Syntax:** :Instrument:GetFormPos? FormID

**Command Argument(s):** FormID <int>

**Response Syntax:** [:Instrument:GetFormPos] Pos

**Response Argument(s):** Pos <doublearray>

**Example:** :Instrument:GetFormPos? 5  
[:Instrument:GetFormPos] 22,62

**Description:** Returns the X,Y position of the form with the specified FormID. (0,0) is the top left corner of the screen. Increasing X moves to the right while increasing Y moves towards the bottom of the screen.

## GetFormSize?

*Command Syntax:* :Instrument:GetFormSize? FormID

*Command Argument(s):* FormID <int>

*Response Syntax:* [:Instrument:GetFormSize] Size

*Response Argument(s):* Size <doublearray>

*Example:* :Instrument:GetFormSize? 5  
[:Instrument:GetFormSize] "303,338"

*Description:* Returns the size in pixels (X,Y) of the specified form.

## GetFormState?

*Command Syntax:* :Instrument:GetFormState? FormID

*Command Argument(s):* FormID <int>

*Response Syntax:* [:Instrument:GetFormState] State

*Response Argument(s):* State <int> {fsNormal=0 | fsMinimized=1 | fsMaximized=2}

*Example:* :Instrument:GetFormState? Value  
[:Instrument:GetFormState] fsNormal

*Description:* Returns the state of the form with the given FormID.

## GetFormTitle?

*Command Syntax:* :Instrument:GetFormTitle? FormID

*Command Argument(s):* FormID <int>

*Response Syntax:* [:Instrument:GetFormTitle] Title

*Response Argument(s):* Title <string>

*Example:* :Instrument:GetFormTitle? 5  
[:Instrument:GetFormTitle] "A0: Time Domain Detector"

*Description:* Returns the title string of the specified form.

## GetMainFormSize?

*Command Syntax:* :Instrument:GetMainFormSize?

*Command Argument(s):* None

*Response Syntax:* [:Instrument:GetMainFormSize] Size

*Response Argument(s):* Size <doublearray>

*Example:* :Instrument:GetMainFormSize?  
[:Instrument:GetMainFormSize] "612,401"

*Description:* Returns the size (X,Y), in pixels, of the main form.

## MoveForm

**Command Syntax:** :Instrument:MoveForm *FormID*, *Page*

**Command Argument(s):** *FormID* <int>  
*Page* <int>

**Example:** :Instrument:MoveForm 5, 3

**Description:** Moves the form with the given FormID to the specified page (1-7) on the page control.

## MoveSelectedForm

**Command Syntax:** :Instrument:MoveSelectedForm *Page*

**Command Argument(s):** *Page* <int>

**Example:** :Instrument:MoveSelectedForm 3

**Description:** Moves the currently selected form to the specified page (1-7) on the page control.

## SelectedForm?

**Command Syntax:** :Instrument:SelectedForm?

**Command Argument(s):** None

**Response Syntax:** [:Instrument:SelectedForm] *FormID*

**Response Argument(s):** *FormID* <int>

**Example:** :Instrument:SelectedForm?  
[:Instrument:SelectedForm] 5

**Description:** Returns the FormID of the currently selected form.

## SelectForm

**Command Syntax:** :Instrument:SelectForm *FormID*

**Command Argument(s):** *FormID* <int>

**Example:** :Instrument:SelectForm 5

**Description:** Selects the form with the specified FormID.

## TileForms

**Command Syntax:** :Instrument:TileForms

**Command Argument(s):** None

**Example:** :Instrument:TileForms

**Description:** Tiles the forms on the current page of the page control.

## SetFormPos

**Command Syntax:** :Instrument:SetFormPos *FormID*, *Pos*

**Command Argument(s):** *FormID* <int>  
*Pos* <doublearray>

**Example:** :Instrument:SetFormPos 20, "303,40"

**Description:** Sets the position of the form with the given FormID to the X-Y position specified in the array Pos.

## SetFormSize

**Command Syntax:** :Instrument:SetFormSize *FormID*, *Size*

**Command Argument(s):** *FormID* <int>  
*Size* <doublearray>

**Example:** :Instrument:SetFormSize 25, "200,100"

**Description:** Sets the size (in pixels) of the form with the given FormID to the X-Y size specified in the Size array. Note that only re-sizable forms can be sized with this command. Specifying a FormID corresponding to a non-resizable form, such as a panel, will result in an execution error.

## SetFormState

**Command Syntax:** :Instrument:SetFormState *FormID*, *State*

**Command Argument(s):** *FormID* <int>  
*State* <int> {fsNormal=0 | fsMinimized=1 | fsMaximized=2}

**Example:** :Instrument:SetFormState Value, fsNormal

**Description:** Sets the state (Normal, minimized, maximized) of the form with the given FormID.

## Measurement Commands

### LastMeas?

**Command Syntax:** :Instrument>LastMeas? *MeasID*, *UnitStr*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0lmd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 |

msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 |  
msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 |  
msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 |  
msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 |  
msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 |  
msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 |  
msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |  
msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 |  
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |  
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |  
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |  
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |  
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |  
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |  
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |  
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |  
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |  
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |  
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |  
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |  
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |

```
msSweep5=5005}
UnitStr <string>
```

**Response Syntax:** [:Instrument:LastMeas] Meas

**Response Argument(s):** Meas <double>

**Example:** :Instrument:LastMeas? msAnlgFreqA, Hz  
[:Instrument:LastMeas] 1000

**Description:** Returns the last value of the specified scalar measurement. The measurement specified must be a scalar, i.e. a single value. To query vector measurements use the LastVectorMeas? command.

Note that the analyzers corresponding to the specified measurement must be configured for that measurement. For instance, asking for the last value of msA1FFT2spectrumA if A1 is not configured as FFT2 will result in an execution error.

## LastVectorBinMeas?

**Command Syntax:** :Instrument:LastVectorBinMeas? MeasID, BinIndex, UnitStr

**Command Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTtrippleA=1250 | msA0MTtrippleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |



```

msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
BinIndex <int>
UnitStr <string>

```

**Response Syntax:** [:Instrument:LastVectorBinMeas] Meas

**Response Argument(s):** Meas <double>

**Example:** :Instrument:LastVectorBinMeas? msA1FFTspectrum, 10, "Vp"  
**[ :Instrument:LastVectorBinMeas ] 2.3099E-5**

**Description:** Returns the last value of a particular bin of a vector measurement. Note that the analyzers corresponding to the specified measurement must be configured for that measurement. For instance, asking for the last value of msA1FFT2spectrumA if A1 is not configured as FFT2 will result in an execution error.

## LastVectorBinXMeas?

**Command Syntax:** :Instrument:LastVectorBinXMeas? MeasID, BinIndex, UnitStr

**Command Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 |

msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
 msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 |  
 msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 |  
 msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 |  
 msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 |  
 msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 |  
 msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 |  
 msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 |  
 msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 |  
 msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 |  
 msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 |  
 msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 |  
 msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 |  
 msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 |  
 msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 |  
 msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |  
 msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
 msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
 msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
 msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
 msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
 msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
 msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
 msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
 msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
 msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
 msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
 msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |  
 msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
 msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
 msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |  
 msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |  
 msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
 msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
 msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
 msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
 msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
 msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
 msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
 msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
 msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
 msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
 msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
 msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
 msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
 msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
 msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
 msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
 msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |  
 msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
 msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
 msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
 msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
 msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
 msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |

```

msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
BinIndex <int>
UnitStr <string>

```

**Response Syntax:** [:Instrument:LastVectorBinXMeas] Meas

**Response Argument(s):** Meas <double>

**Example:** :Instrument:LastVectorBinXMeas? msA1FFTspectrum, 10, "Hz  
[:Instrument:LastVectorBinXMeas] 2500

**Description:** Queries the X-axis value corresponding the the specified bin of the specified measurement.

## LastVectorMeas?

**Command Syntax:** :Instrument:LastVectorMeas? MeasID, UnitStr

**Command Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |

```

msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTpectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2pectrumA=2122 |
msA1FFT2pectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTpectrumA=2202 | msA1MTpectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
UnitStr <string>

```

**Response Syntax:** [:Instrument:LastVectorMeas] *Vector*

**Response Argument(s):** *Vector* <doublearray>

**Example:** :Instrument:LastVectorMeas? msA1FFTtimeRec, "Vp"  
 [:Instrument:LastVectorMeas] "-1.68,-1.79,-1.72,-1.75,-1

**Description:** Returns an entire array corresponding to the last value of a vector measurement.

## LastVectorXMeas?

**Command Syntax:** :Instrument:LastVectorXMeas? *MeasID*, *UnitStr*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 |  
 msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 |  
 msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 |  
 msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 |  
 msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 |

msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
 msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 |  
 msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 |  
 msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 |  
 msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 |  
 msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 |  
 msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 |  
 msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 |  
 msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 |  
 msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 |  
 msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 |  
 msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 |  
 msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 |  
 msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 |  
 msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 |  
 msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |  
 msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
 msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
 msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
 msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
 msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
 msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
 msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
 msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
 msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
 msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
 msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
 msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |  
 msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
 msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
 msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |  
 msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |  
 msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
 msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
 msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
 msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
 msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
 msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
 msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
 msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
 msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
 msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
 msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
 msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
 msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
 msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
 msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
 msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
 msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |  
 msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
 msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
 msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
 msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
 msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
 msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |

```

msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
UnitStr <string>

```

**Response Syntax:** [:Instrument:LastVectorXMeas] *Vector*

**Response Argument(s):** *Vector* <doublearray>

**Example:** :Instrument:LastVectorXMeas? msA1FFTspectrum, "Hz"  
[:Instrument:LastVectorXMeas] "0,250,500,750,1000,1250,1"

**Description:** Returns an entire array corresponding to the X-axis of a vector measurement.

## MeasCount?

**Command Syntax:** :Instrument:MeasCount?

**Command Argument(s):** None

**Response Syntax:** [:Instrument:MeasCount] *Count*

**Response Argument(s):** *Count* <int>

**Example:** :Instrument:MeasCount?  
[:Instrument:MeasCount] 22

**Description:** Queries the number of active measurements currently being made by SR1.

## MeasInfo?

**Command Syntax:** :Instrument:MeasInfo? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MThdBinsA=1206 | msA0MThdBinsB=1207 |

msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTdA=1248 |  
msA0MTdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |  
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2timeRecA=2111 |  
msA1FFT2timeRecB=2112 | msA1FFT2timeRecC=2113 | msA1FFT2timeRecA=2120 |  
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |  
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |  
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |  
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |  
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |  
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |  
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTdA=2248 |  
msA1MTdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |  
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |  
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |  
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |  
msSweep5=5005}

**Response Syntax:** [:Instrument:MeasInfo] Info

**Response Argument(s):** Info <string>

**Example:** :Instrument:MeasInfo? msA1FFT2spectrum

**[:Instrument:MeasInfo] "A1:FFT1:Power Spectrum A (vector**

**Description:** Returns a string describing the specified measurement.

## MeasItem?

**Command Syntax:** :Instrument:MeasItem? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:Instrument:MeasItem] MeasID

**Response Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 | msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 | msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 | msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |



```

msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Instrument:MeasItem? 10  
[:Instrument:MeasItem] MSA1LEVELA

**Description:** Returns the measurement ID for  $i^{\text{th}}$  measurement being made by the instrument. The first measurement corresponds to  $i=0$ . The total number of measurements can be determined using the MeasCount? command.

## MeasUnit?

**Command Syntax:** :Instrument:MeasUnit? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |

```

msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTdA=1248 |
msA0MTdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTdA=2248 |
msA1MTdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Instrument:MeasUnit] UnitStr

**Response Argument(s):** UnitStr <string>

**Example:** :Instrument:MeasUnit? msA1FFT2spectrum  
[:Instrument:MeasUnit] "Vp"

**Description:** Returns a string corresponding to the default units for measurement with the specified MeasID.

## XUnitStr?

**Command Syntax:** :Instrument:XUnitStr? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2pectrumA=1122 | msA0FFT2pectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTpectrumA=1202 | msA0MTpectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTpectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2pectrumA=2122 | msA1FFT2pectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 | msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 | msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |

```

msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Instrument:XUnitStr] UnitStr

**Response Argument(s):** UnitStr <string>

**Example:** :Instrument:XUnitStr? msA1FFTpectrum  
[:Instrument:XUnitStr] Hz

**Description:** Queries the X-units of an active vector measurement.

## SettlerIndex?

**Command Syntax:** :Instrument:SettlerIndex? MeasID

**Command Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2pectrumA=1122 | msA0FFT2pectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTpectrumA=1202 | msA0MTpectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |

```

msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Instrument:SettlerIndex] SettlerIndex

**Response Argument(s):** SettlerIndex <int>

**Example:** :Instrument:SettlerIndex? msNull  
[:Instrument:SettlerIndex] Value

**Description:** Returns the index of the settler corresponding to the specified measurement. The properties of the settler can then be accessed using the commands in [.Sweep: Settling\(i\)](#).

## MeasGetStorageDepth?

**Command Syntax:** :Instrument:MeasGetStorageDepth? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2pectrumA=1122 | msA0FFT2pectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTpectrumA=1202 | msA0MTpectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTpectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2pectrumA=2122 | msA1FFT2pectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 | msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 | msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |

```

msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Instrument:MeasGetStorageDepth] *Depth*

**Response Argument(s):** *Depth* <int>

**Example:** :Instrument:MeasGetStorageDepth? msAnlgFreqA  
[:Instrument:MeasGetStorageDepth] 256

**Description:** Queries the maximum storage depth for the measurement specified.

## MeasSetStorageDepth

**Command Syntax:** :Instrument:MeasSetStorageDepth *MeasID*, *Depth*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |

```

msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
Depth <int>

```

**Example:** :Instrument:MeasSetStorageDepth msA1FFT2spectrum, 5

**Description:** Sets the maximum storage depth for the specified measurement. In free run mode SR1 maintains a stack of this many values of the measurement.

## MeasGetNumStoredData?

**Command Syntax:** :Instrument:MeasGetNumStoredData? MeasID

**Command Argument(s):** MeasID <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 |



msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 |  
 msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 |  
 msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 |  
 msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 |  
 msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 |  
 msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 |  
 msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 |  
 msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 |  
 msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 |  
 msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 |  
 msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 |  
 msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 |  
 msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 |  
 msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 |  
 msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 |  
 msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 |  
 msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 |  
 msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 |  
 msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 |  
 msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
 msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
 msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
 msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
 msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
 msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
 msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
 msA0MTxtalkVsFreqA=1230 | msA0MTxtalkVsFreqB=1231 |  
 msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
 msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
 msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
 msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |  
 msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
 msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
 msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |  
 msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |  
 msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
 msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
 msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
 msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
 msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
 msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
 msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
 msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
 msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
 msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
 msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
 msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
 msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
 msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
 msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
 msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
 msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |  
 msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
 msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
 msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |

```

msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Instrument:MeasGetNumStoredData] *N*

**Response Argument(s):** *N* <int>

**Example:** :Instrument:MeasGetNumStoredData? msNull  
[:Instrument:MeasGetNumStoredData] Value

**Description:** Queries the current storage depth for the specified measurement.

## StoredVectorMeas?

**Command Syntax:** :Instrument:StoredVectorMeas? *MeasID*, *UnitStr*, *Index*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqA=1230 | msA0MTxtalkVsFreqB=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |

```

msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
UnitStr <string>
Index <int>

```

**Response Syntax:** [:Instrument:StoredVectorMeas] *Vector*

**Response Argument(s):** *Vector* <doublearray>

**Example:** :Instrument:StoredVectorMeas? msA1FFTspectrum, "", 4  
[:Instrument:StoredVectorMeas] 2.3099E-5, 3.8033E-5, 2.

**Description:** Returns the vector corresponding to the Index<sup>th</sup> stored measurement of type MeasID. Index = 0 corresponds to the oldest measurement, increasing index corresponds to more recent measurements.

## User Input Commands

### UserChoice?

**Command Syntax:** :Instrument:UserChoice? Message, Choices, Timeout

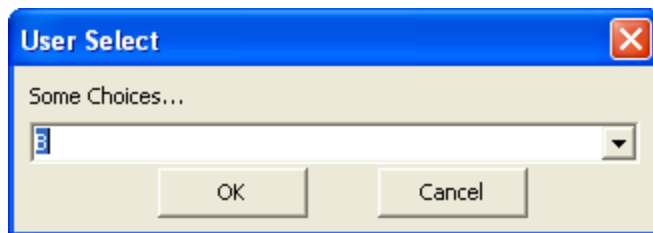
**Command Argument(s):** Message <string>  
Choices <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserChoice] Result

**Response Argument(s):** Result <string>

**Example:** :Instrument:UserChoice? "Some Choices...", "A,B,C", 100  
[:Instrument:UserChoice] B

**Description:** Displays a drop-down user selection window:



The "Message" string is displayed over a dropdown list filled with the choices contained in the "Choices" argument. (Choices contains a string with each choice separated by a comma). The timeout argument specifies how long the window will remain open waiting for a user selection. When the user does select something the query returns a string containing the selected choice (if "OK" was pressed), "-cancelled" (if Cancel was pressed), or "-timedout-" (if the window timed out).

## UserChoiceMulti?

**Command Syntax:** :Instrument:UserChoiceMulti? *Message, Choices, Timeout*

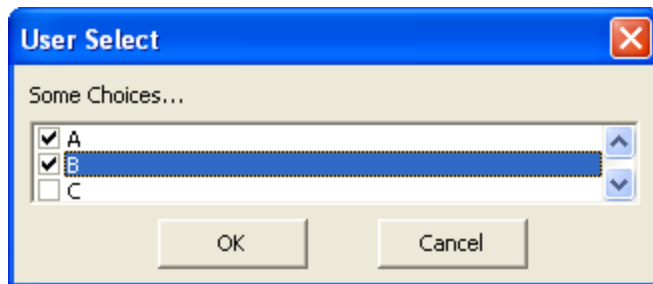
**Command Argument(s):** *Message* <string>  
*Choices* <string>  
*Timeout* <int>

**Response Syntax:** [:Instrument:UserChoiceMulti] *Result*

**Response Argument(s):** *Result* <string>

**Example:** :Instrument:UserChoiceMulti? "Some Choices...", "A,B,C", 1  
[:Instrument:UserChoiceMulti] "A,B"

**Description:** Displays a checkbox user selection window:



The "Message" string is displayed over a list of checkboxes labeled with the choices contained in the "Choices" argument. (Choices contains a string with each choice separated by a comma). The timeout argument specifies how long the window will remain open waiting for a user selection. When the user does select something the query returns a string containing the selected choices (if "OK" was pressed), "-cancelled" (if Cancel was pressed), or "-timedout-" (if the window timed out).

## UserInput?

**Command Syntax:** :Instrument:UserInput? Message, DefaultResult, Timeout

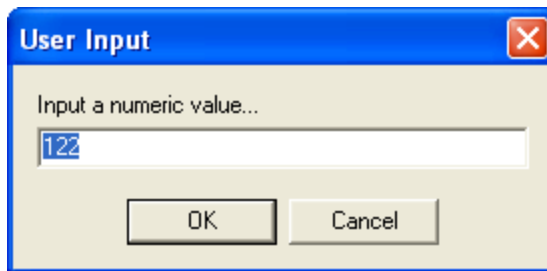
**Command Argument(s):** Message <string>  
DefaultResult <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserInput] Result

**Response Argument(s):** Result <string>

**Example:** :Instrument:UserInput? "Input a numeric value...","default"  
[:Instrument:UserInput] 122

**Description:** Displays a window with a single edit control for user input:



The "Message" string is displayed over an edit control allowing user input. The timeout argument specifies how long the window will remain open waiting for user action. The query returns the string entered by the user (if "OK" was pressed), "-cancelled" (if Cancel was pressed), or "-timedout-" (if the window timed out).

## UserLaunchChoice?

**Command Syntax:** :Instrument:UserLaunchChoice? *Message, Buttons, Descriptions, Timeout*

**Command Argument(s):** *Message* <string>  
*Buttons* <string>  
*Descriptions* <string>  
*Timeout* <int>

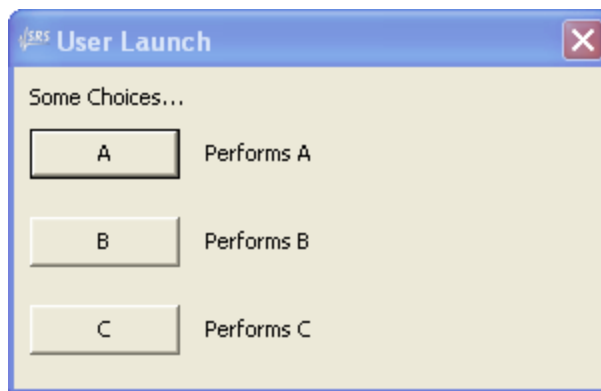
**Response Syntax:** [:Instrument:UserLaunchChoice] *Result*

**Response Argument(s):** *Result* <string>

**Example:** :Instrument:UserLaunchChoice? "Some Choices...", "A,B,C"  
 "Performs A,Performs A,Performs A", 100

**[:Instrument:UserLaunchChoice] A**

**Description:** Displays a window with a series of labeled buttons:



The "Message" string is displayed over a list of buttons. The "Buttons" argument is a comma separated list of labels for the buttons. The "Descriptions" argument lists the labels next to the buttons. The timeout argument specifies how long the window will remain open waiting for a user selection. When the user presses a button the query returns a string containing the selected choice (if a button was pressed), "-cancelled" (if the window was closed), or "-timedout-" (if the window timed out).

## UserLoadFile?

**Command Syntax:** :Instrument:UserLoadFile? Message, Timeout

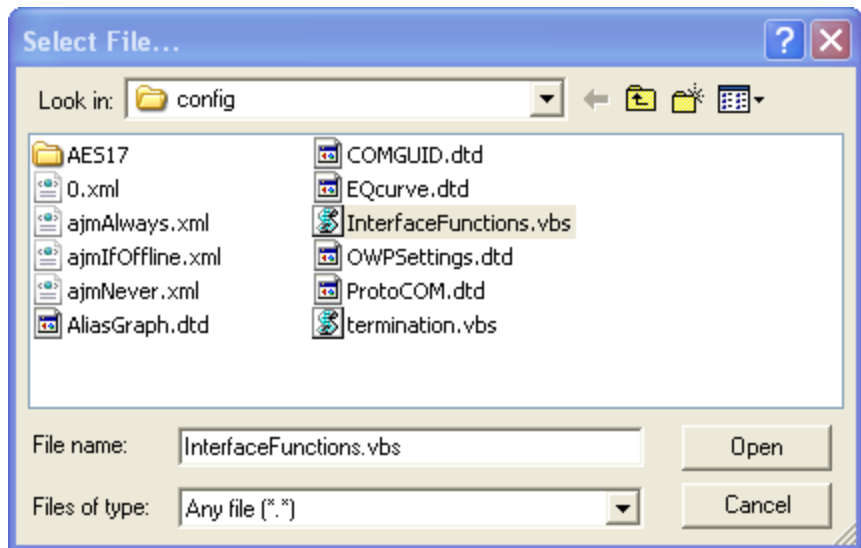
**Command Argument(s):** Message <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserLoadFile] Result

**Response Argument(s):** Result <string>

**Example:** :Instrument:UserLoadFile? "Select File...", 100  
[:Instrument:UserLoadFile] C:\Build\user\config\Interfac

**Description:** Displays a file load dialog box:



The query returns the string containing the path and filename of the selected file, or "-cancelled-".



## UserMessage?

**Command Syntax:** :Instrument:UserMessage? Message, Timeout

**Command Argument(s):** Message <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserMessage] Result

**Response Argument(s):** Result <int> {umTimedout=-1 | umOK=1}

**Example:** :Instrument:UserMessage? "User Message",100  
[:Instrument:UserMessage] umTimedout

**Description:** Displays a window with a user message:



The query returns either umTimedout or umOK depending on whether the user presses "OK" before the timeout interval expires.

## UserOKCancel?

**Command Syntax:** :Instrument:UserOKCancel? Message, Timeout

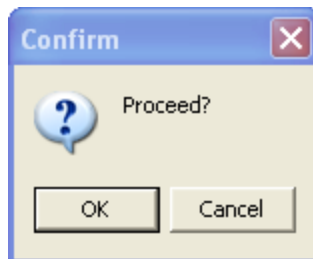
**Command Argument(s):** Message <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserOKCancel] Result

**Response Argument(s):** Result <int> {ocTimedout=-1 | ocCancel=0 | ocOK=1}

**Example:** :Instrument:UserOKCancel? Value, Value  
[:Instrument:UserOKCancel] ocTimedout

**Description:** Displays a window with a user message and an OK and Cancel button.



The query returns ocTimedout, ocCancel, or ocOK depending on the user action.

## UserSaveFile?

**Command Syntax:** :Instrument:UserSaveFile? Message, Timeout

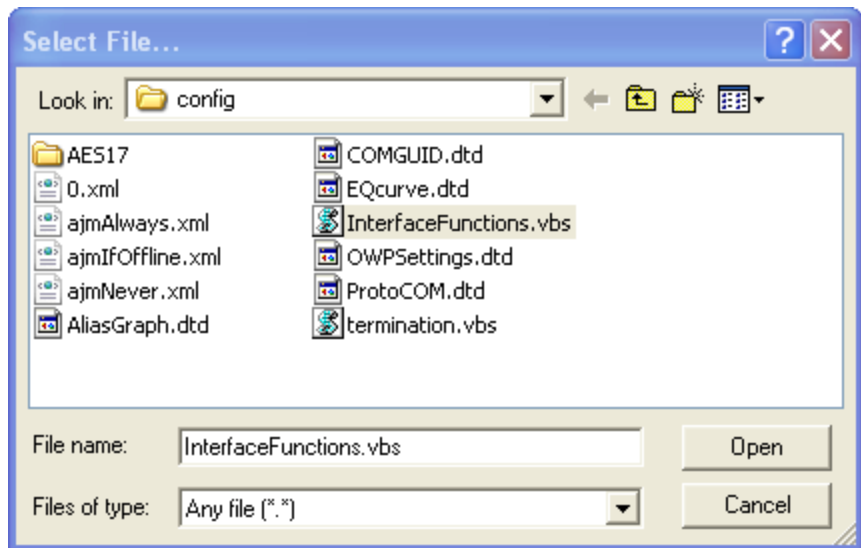
**Command Argument(s):** Message <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserSaveFile] Result

**Response Argument(s):** Result <string>

**Example:** :Instrument:UserSaveFile? Value, Value  
[:Instrument:UserSaveFile] Value

**Description:** Displays a file save dialog box:



The query returns the string containing the path and filename of the selected file, or "-cancelled-".

## UserYesNoCancel?

**Command Syntax:** :Instrument:UserYesNoCancel? Message, Timeout

**Command Argument(s):** Message <string>  
Timeout <int>

**Response Syntax:** [:Instrument:UserYesNoCancel] Result

**Response Argument(s):** Result <int> {yncTimedout=-1 | yncNo=0 | yncYes=1 | yncCancel=2}

**Example:** :Instrument:UserYesNoCancel? Value, Value  
[:Instrument:UserYesNoCancel] yncTimedout

**Description:** Displays a window with a user message, a "Yes", "No", and "Cancel" button. The query returns the value corresponding to the user action.

## Wait

**Command Syntax:** :Instrument:Wait *Duration*

**Command Argument(s):** *Duration* <int>

**Example:** :Instrument:Wait 1000

**Description:** Causes a delay for *Duration* milliseconds

## Misc. Instrument Commands

### Load

**Command Syntax:** :Instrument:Load *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Instrument:Load "MyConfig.xml"

**Description:** Loads the specified SR1 configuration file.

### LoadPartial

**Command Syntax:** :Instrument:LoadPartial *FileName, Mask*

**Command Argument(s):** *FileName* <string>

*Mask* <int> {fmAll=1 | fmAnlgGen=1 | fmDigGen=2 | fmMultitoneConfig=4 |  
fmAlyzr0=8 | fmAlyzr1=16 | fmDigitizer=32 | fmAnlgInput=64 |  
fmAlyzrReferences=128 | fmClockRef=256 | fmDigIO=512 | fmDisplays=1024 |  
fmSweep=2048 | fmEventMgr=4096 | fmRemoteIfc=8192 | fmScripting=16384 |  
fmQuickMeas=32768 | fmSwitcher=65536 | fmMonitor=131072}

**Example:** :Instrument:LoadPartial MyConfig, fmAnlgGen

**Description:** Loads only the specified portions from the configuration. Each portion of the instrument is represented by a single bit in the "Mask" argument. For instance, Mask = 257 corresponds to the analog generator plus the clock reference portions of the instrument.

### Beep

**Command Syntax:** :Instrument:Beep

**Command Argument(s):** None

**Example:** :Instrument:Beep

**Description:** Makes a "beep" sound.

### Reset

**Command Syntax:** :Instrument:Reset

**Command Argument(s):** None

**Example:** :Instrument:Reset

**Description:** Resets the instrument to its default state.

## Save

**Command Syntax:** :Instrument:Save *FileName*, *GraphData*

**Command Argument(s):** *FileName* <string>  
*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Instrument:Save Value, gdNeverSave

**Description:** Saves the entire instrument configuration. The *GraphData* argument specifies whether to include graph data in the configuration file.

## SavePartial

**Command Syntax:** :Instrument:SavePartial *FileName*, *Mask*, *GraphData*

**Command Argument(s):** *FileName* <string>  
*Mask* <int> {fmAll=1 | fmAnlgGen=1 | fmDigGen=2 | fmMultitoneConfig=4 |  
 fmAnlyzr0=8 | fmAnlyzr1=16 | fmDigitizer=32 | fmAnlgInput=64 |  
 fmAnlyzrReferences=128 | fmClockRef=256 | fmDigIO=512 | fmDisplays=1024 |  
 fmSweep=2048 | fmEventManager=4096 | fmRemoteIfc=8192 | fmScripting=16384 |  
 fmQuickMeas=32768 | fmSwitcher=65536 | fmMonitor=131072}  
*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Instrument:SavePartial Value, fmAll, gdNeverSave

**Description:** Saves a portion of the current configuration to an file. Each portion of the instrument is represented by a single bit in the "Mask" argument. For instance, Mask = 257 corresponds to the analog generator plus the clock reference portions of the instrument. The *GraphData* argument specifies whether to include graph data in the configuration file.

## SerialNo?

**Command Syntax:** :Instrument:SerialNo?

**Command Argument(s):** None

**Response Syntax:** [:Instrument:SerialNo] *SN*

**Response Argument(s):** *SN* <string>

**Example:** :Instrument:SerialNo?  
 [:Instrument:SerialNo] Value

**Description:** Returns the serial number of the instrument.

## Version?

**Command Syntax:** :Instrument:Version?

**Command Argument(s):** None

**Response Syntax:** [:Instrument:Version] *Ver*

**Response Argument(s):** *Ver* <string>

**Example:** :Instrument:Version?  
 [:Instrument:Version] 1.2.2.0

**Description:** Returns the version string corresponding to the instrument software.

## CaptureScreen

**Command Syntax:** :Instrument:CaptureScreen *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Instrument:CaptureScreen "Image.bmp"

**Description:** Takes a screenshot of the instrument, and saves it to the specified file.

## TransferFile?

**Command Syntax:** :Instrument:TransferFile? *FileName*

**Command Argument(s):** *FileName* <string>

**Response Syntax:** [:Instrument:TransferFile] *Data*

**Response Argument(s):** *Data* <string>

**Example:** :Instrument:TransferFile? "Image.bmp"

**[ :Instrument:TransferFile ] #42048...binary.file.data...**

**Description:** Returns the contents of the specified file in definite length arbitrary block format. The data returned may contain binary data (i.e. each character may have a numerical value of 0-255). Note that the last character returned is always a linefeed, as required by IEEE-488.2; it is **not** part of the file data.

### 2.3.14 Preferences

<b>Object:</b>	<b>:Preferences</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Preferences panel.

## General Preference Commands



### StartupConfig?

**Command Syntax:** :Preferences:StartupConfig?

**Command Argument(s):**

**Response Syntax:** [:Preferences:StartupConfig] Value

**Response Argument(s):** Value <int> {scSR1Default=0 | scUserDefault=1 | scLastSaved=2}

**Example:** :Preferences:StartupConfig?

**[[:Preferences:StartupConfig] scSR1Default**

**Related Command(s):** StartupConfig

**Description:** Queries the startup configuration mode.

### StartupConfig

**Command Syntax:** :Preferences:StartupConfig Value [, AllowCoercion]

**Command Argument(s):** Value <int> {scSR1Default=0 | scUserDefault=1 | scLastSaved=2}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:StartupConfig scSR1Default

**Related Command(s):** StartupConfig?

**Description:** Sets the startup configuration.

## UserDefaultConfigFile?

**Command Syntax:** :Preferences:UserDefaultConfigFile?

**Command Argument(s):**

**Response Syntax:** [:Preferences:UserDefaultConfigFile] Value

**Response Argument(s):** Value <string>

**Example:** :Preferences:UserDefaultConfigFile?

**[ :Preferences:UserDefaultConfigFile ] MyStartup.XML**

**Related Command(s):** UserDefaultConfigFile

**Description:** Queries the startup configuration file used in "User Default" mode.

## UserDefaultConfigFile

**Command Syntax:** :Preferences:UserDefaultConfigFile Value [, AllowCoercion]

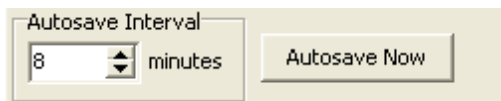
**Command Argument(s):** Value <string>

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Preferences:UserDefaultConfigFile "MyStarup.XML"

**Related Command(s):** UserDefaultConfigFile?

**Description:** Sets the startup configuration file used in "User Default" mode.



## AutosaveInterval?

**Command Syntax:** :Preferences:AutosaveInterval?

**Command Argument(s):**

**Response Syntax:** [:Preferences:AutosaveInterval] Value

**Response Argument(s):** Value <int>

**Example:** :Preferences:AutosaveInterval?

**[ :Preferences:AutosaveInterval ] Value**

**Related Command(s):** AutosaveInterval

**Description:** Queries the time interval between autosaves.

## AutosaveInterval

**Command Syntax:** :Preferences:AutosaveInterval Value [, AllowCoercion]

**Command Argument(s):** Value <int>

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Preferences:AutosaveInterval Value

**Related Command(s):** AutosaveInterval?

**Description:** Sets the time interval between autosaves.

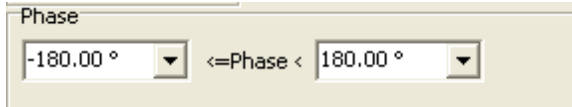
## AutosaveNow

**Command Syntax:** :Preferences:AutosaveNow *SaveDuringSweep*

**Command Argument(s):** *SaveDuringSweep* <bool> {False=0 | True=1}

**Example:** :Preferences:AutosaveNow False

**Description:** Saves the current instrument configuration to the autosave file.



## PhaseMax?

**Command Syntax:** :Preferences:PhaseMax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Preferences:PhaseMax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Preferences:PhaseMax?

[ :Preferences:PhaseMax ] 180 °

**Related Command(s):** PhaseMax

**Description:** Queries the maximum value of the phase measurement range.

## PhaseMax

**Command Syntax:** :Preferences:PhaseMax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:PhaseMax 180 deg

**Related Command(s):** PhaseMax?

**Description:** Sets the maximum value of the phase measurement range

## PhaseMin?

**Command Syntax:** :Preferences:PhaseMin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Preferences:PhaseMin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Preferences:PhaseMin?

[ :Preferences:PhaseMin ] *Value*

**Related Command(s):** PhaseMin

**Description:** Queries the minimum value of the phase measurement range.



## PhaseMin

**Command Syntax:** `:Preferences:PhaseMin Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:PhaseMin Value`

**Related Command(s):** PhaseMin?

**Description:** Sets the minimum value of the phase measurement range.



## SignalInit?

**Command Syntax:** `:Preferences:SignalInit?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:SignalInit] Value`

**Response Argument(s):** `Value <int> {siSR1Default=0 | siLastUsed=1 | siLastUsedExclAmp=2 | siLastUsedSigOff=3}`

**Example:** `:Preferences:SignalInit?`

`[:Preferences:SignalInit] siSR1Default`

**Related Command(s):** SignalInit

**Description:** Queries the generator signal initialization method.

## SignalInit

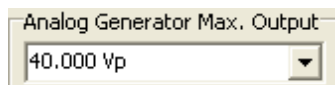
**Command Syntax:** `:Preferences:SignalInit Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {siSR1Default=0 | siLastUsed=1 | siLastUsedExclAmp=2 | siLastUsedSigOff=3}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:SignalInit siSR1Default`

**Related Command(s):** SignalInit?

**Description:** Sets the generator signal initialization method.



## AnlgGenMaxVolt?

**Command Syntax:** :Preferences:AnlgGenMaxVolt? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Preferences:AnlgGenMaxVolt] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Preferences:AnlgGenMaxVolt?  
[:Preferences:AnlgGenMaxVolt] 40 VP

**Related Command(s):** AnlgGenMaxVolt

**Description:** Queries the maximum analog generator output voltage.

## AnlgGenMaxVolt

**Command Syntax:** :Preferences:AnlgGenMaxVolt *Value* [, *AllowCoercion*]

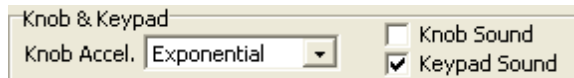
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:AnlgGenMaxVolt 40 Vp

**Related Command(s):** AnlgGenMaxVolt?

**Description:** Sets the maximum analog generator output voltage



## KeypadSound?

**Command Syntax:** :Preferences:KeypadSound?

**Command Argument(s):**

**Response Syntax:** [:Preferences:KeypadSound] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:KeypadSound?  
[:Preferences:KeypadSound] False

**Related Command(s):** KeypadSound

**Description:** Queries the on/off status of the sound made when a keypad key is pressed.

## KeypadSound

**Command Syntax:** :Preferences:KeypadSound *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:KeypadSound False

**Related Command(s):** KeypadSound?

**Description:** Sets the on/off status of the sound made when a keypad key is pressed.

## KnobAccel?

**Command Syntax:** `:Preferences:KnobAccel?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:KnobAccel] Value`

**Response Argument(s):** `Value <int> {kaExponential=0 | kaPowerLaw=1 | kaCursorPos=2}`

**Example:** `:Preferences:KnobAccel?`

`[ :Preferences:KnobAccel ] kaExponential`

**Related Command(s):** KnobAccel

**Description:** Queries the knob acceleration algorithm.

## KnobAccel

**Command Syntax:** `:Preferences:KnobAccel Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {kaExponential=0 | kaPowerLaw=1 | kaCursorPos=2}`

`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:KnobAccel kaExponential`

**Related Command(s):** KnobAccel?

**Description:** Sets the knob acceleration algorithm.

## KnobSound?

**Command Syntax:** `:Preferences:KnobSound?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:KnobSound] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Preferences:KnobSound?`

`[ :Preferences:KnobSound ] False`

**Related Command(s):** KnobSound

**Description:** Queries the on/off status of the sound made when the knob is turned.

## KnobSound

**Command Syntax:** `:Preferences:KnobSound Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`

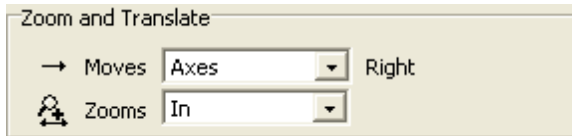
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:KnobSound False`

**Related Command(s):** KnobSound?

**Description:** Sets the on/off status of the sound made when the knob is turned.

## Display Preference Commands



### ShiftMode?

**Command Syntax:** :Preferences:ShiftMode?

**Command Argument(s):**

**Response Syntax:** [:Preferences:ShiftMode] Value

**Response Argument(s):** Value <int> {spShiftData=0 | spShiftAxes=1}

**Example:** :Preferences:ShiftMode?

**[ :Preferences:ShiftMode ] spShiftData**

**Related Command(s):** ShiftMode

**Description:** Queries the sense of the translate scaling control (moves data vs. moves axes).

### ShiftMode

**Command Syntax:** :Preferences:ShiftMode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {spShiftData=0 | spShiftAxes=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:ShiftMode spShiftData

**Related Command(s):** ShiftMode?

**Description:** Sets the sense of the translate scaling control (moves data vs. moves axes)

### ZoomMode?

**Command Syntax:** :Preferences:ZoomMode?

**Command Argument(s):**

**Response Syntax:** [:Preferences:ZoomMode] Value

**Response Argument(s):** Value <int> {zpZoomData=0 | zpZoomAxes=1}

**Example:** :Preferences:ZoomMode?

**[ :Preferences:ZoomMode ] zpZoomData**

**Related Command(s):** ZoomMode

**Description:** Queries the sense of the zoom scaling controls (moves data vs. moves axes).

## ZoomMode

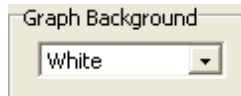
**Command Syntax:** :Preferences:ZoomMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {zpZoomData=0 | zpZoomAxes=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:ZoomMode zpZoomData

**Related Command(s):** ZoomMode?

**Description:** Sets the sense of the zoom scaling controls (moves data vs. moves axes)..



## GraphBackgroundColor?

**Command Syntax:** :Preferences:GraphBackgroundColor?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GraphBackgroundColor] *Value*

**Response Argument(s):** *Value* <int> {bcWhite=0 | bcBlack=1}

**Example:** :Preferences:GraphBackgroundColor?

**[ : Preferences : GraphBackgroundColor ] bcWhite**

**Related Command(s):** GraphBackgroundColor

**Description:** Queries the default graph background color (black or white).

## GraphBackgroundColor

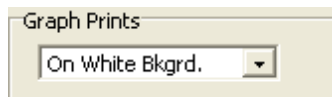
**Command Syntax:** :Preferences:GraphBackgroundColor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {bcWhite=0 | bcBlack=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:GraphBackgroundColor bcWhite

**Related Command(s):** GraphBackgroundColor?

**Description:** Sets the default graph background color (black or white).



## GraphPrint?

**Command Syntax:** :Preferences:GraphPrint?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GraphPrint] Value

**Response Argument(s):** Value <int> {pbAsDisplayed=0|pbWhiteBackground=1}

**Example:** :Preferences:GraphPrint?

**[ :Preferences:GraphPrint ] pbAsDisplayed**

**Related Command(s):** GraphPrint

**Description:** Queries whether graphs print with a white or black background.

## GraphPrint

**Command Syntax:** :Preferences:GraphPrint Value [, AllowCoercion]

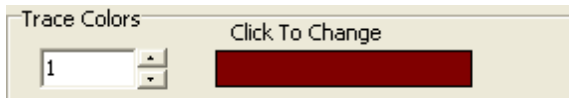
**Command Argument(s):** Value <int> {pbAsDisplayed=0|pbWhiteBackground=1}

AllowCoercion <bool> {False=0|True=1}

**Example:** :Preferences:GraphPrint pbAsDisplayed

**Related Command(s):** GraphPrint?

**Description:** Sets whether graphs print with a white or black background.



## GetTraceColor?

**Command Syntax:** :Preferences:GetTraceColor? Index, BGColor

**Command Argument(s):** Index <int>

BGColor <int> {bcWhite=0|bcBlack=1}

**Response Syntax:** [:Preferences:GetTraceColor] Color

**Response Argument(s):** Color <int>

**Example:** :Preferences:GetTraceColor? 1, bcWhite

**[ :Preferences:GetTraceColor ] 4532**

**Description:** Returns the color used for the Index<sup>th</sup> trace with the specified graph background color. (The set of colors is different depending on whether a white or black background is used.)

## SetTraceColor

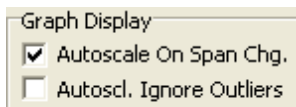
**Command Syntax:** :Preferences:SetTraceColor Index, Color

**Command Argument(s):** Index <int>

Color <int>

**Example:** :Preferences:SetTraceColor 1, 4532

**Description:** Sets the color used for the Index<sup>th</sup> trace with the current graph background color. (The set of colors is different depending on whether a white or black background is used.)



## AutoSpanFFT?

**Command Syntax:** :Preferences:AutoSpanFFT?

**Command Argument(s):**

**Response Syntax:** [:Preferences:AutoSpanFFT] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:AutoSpanFFT?  
[:Preferences:AutoSpanFFT] **False**

**Related Command(s):** AutoSpanFFT

**Description:** Queries whether traces displaying live FFT data will automatically scale when the FFT analysis range is changed.

## AutoSpanFFT

**Command Syntax:** :Preferences:AutoSpanFFT *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:AutoSpanFFT **False**

**Related Command(s):** AutoSpanFFT?

**Description:** Sets whether traces displaying live FFT data will automatically scale when the FFT analysis range is changed.

## GraphAutosclIgnoreOutliers?

**Command Syntax:** :Preferences:GraphAutosclIgnoreOutliers?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GraphAutosclIgnoreOutliers] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:GraphAutosclIgnoreOutliers?  
[:Preferences:GraphAutosclIgnoreOutliers] **False**

**Related Command(s):** GraphAutosclIgnoreOutliers

**Description:** Queries whether the graph autoscale algorithm will ignore isolated outlier points.

## GraphAutosclIgnoreOutliers

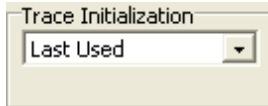
**Command Syntax:** `:Preferences:GraphAutosclIgnoreOutliers Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:GraphAutosclIgnoreOutliers False`

**Related Command(s):** `GraphAutosclIgnoreOutliers?`

**Description:** Sets whether the graph autoscale algorithm will ignore isolated outlier points.



## TraceInit?

**Command Syntax:** `:Preferences:TraceInit?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:TraceInit] Value`

**Response Argument(s):** `Value <int> {tiSR1Default=0 | tiLastUsed=1}`

**Example:** `:Preferences:TraceInit?`

`[:Preferences:TraceInit] tiSR1Default`

**Related Command(s):** `TraceInit`

**Description:** Queries whether new graph traces will be initialized with default scaling values, or values corresponding to the last ones used for a similar trace.

## TraceInit

**Command Syntax:** `:Preferences:TraceInit Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {tiSR1Default=0 | tiLastUsed=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:TraceInit tiSR1Default`

**Related Command(s):** `TraceInit?`

**Description:** Sets whether new graph traces will be initialized with default scaling values, or values corresponding to the last ones used for a similar trace.

## ScreenSize?

**Command Syntax:** `:Preferences:ScreenSize?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:ScreenSize] Value`

**Response Argument(s):** `Value <int> {ss100pct=0 | ss122pct=1 | ss136pct=2 | ss152pct=3}`

**Example:** `:Preferences:ScreenSize?`

`[:Preferences:ScreenSize] ss100pct`

**Related Command(s):** `ScreenSize`

**Description:** Queries the value of the panel size parameter.



## ScreenSize

**Command Syntax:** `:Preferences:ScreenSize Value [, AllowCoercion]`

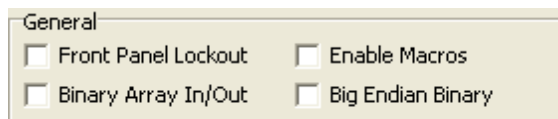
**Command Argument(s):** `Value <int> {ss100pct=0 | ss122pct=1 | ss136pct=2 | ss152pct=3}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:ScreenSize ss100pct`

**Related Command(s):** ScreenSize?

**Description:** Sets the value of the panel size parameter.

## Remote Interface Commands



## RemotePanelLockout?

**Command Syntax:** `:Preferences:RemotePanelLockout?`

**Command Argument(s):** (None)

**Response Syntax:** `[:Preferences:RemotePanelLockout] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Preferences:RemotePanelLockout?`

`[:Preferences:RemotePanelLockout] False`

**Related Command(s):** RemotePanelLockout

**Description:** Queries whether the instrument implements "Remote" mode, i.e. front panel lockout.

## RemotePanelLockout

**Command Syntax:** `:Preferences:RemotePanelLockout Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:RemotePanelLockout False`

**Related Command(s):** RemotePanelLockout?

**Description:** Sets whether the instrument implements "Remote" mode, i.e. front panel lockout.

## RemoteArrayIO?

**Command Syntax:** :Preferences:RemoteArrayIO?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteArrayIO] Value

**Response Argument(s):** Value <int> {baASCII=0 | baBinary=1}

**Example:** :Preferences:RemoteArrayIO?

**[ :Preferences:RemoteArrayIO] baASCII**

**Related Command(s):** RemoteArrayIO

**Description:** Queries whether commands which return arrays will return ascii or binary values.

## RemoteArrayIO

**Command Syntax:** :Preferences:RemoteArrayIO Value [, AllowCoercion]

**Command Argument(s):** Value <int> {baASCII=0 | baBinary=1}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteArrayIO baASCII

**Related Command(s):** RemoteArrayIO?

**Description:** Sets whether commands which return arrays will return ascii or binary values.

## RemoteBinaryEndian?

**Command Syntax:** :Preferences:RemoteBinaryEndian?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteBinaryEndian] Value

**Response Argument(s):** Value <int> {beLittleEndian=0 | beBigEndian=1}

**Example:** :Preferences:RemoteBinaryEndian?

**[ :Preferences:RemoteBinaryEndian] beLittleEndian**

**Related Command(s):** RemoteBinaryEndian

**Description:** Queries the Big/Little Endian format that binary arrays will be sent in.

## RemoteBinaryEndian

**Command Syntax:** :Preferences:RemoteBinaryEndian Value [, AllowCoercion]

**Command Argument(s):** Value <int> {beLittleEndian=0 | beBigEndian=1}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteBinaryEndian beLittleEndian

**Related Command(s):** RemoteBinaryEndian?

**Description:** Sets the Big/Little Endian format that binary arrays will be sent in.

## RemoteEnableMacro?

**Command Syntax:** :Preferences:RemoteEnableMacro?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteEnableMacro] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:RemoteEnableMacro?  
 [:Preferences:RemoteEnableMacro] **False**

**Related Command(s):** RemoteEnableMacro

**Description:** Queries the enabled status of GPIB macros.

## RemoteEnableMacro

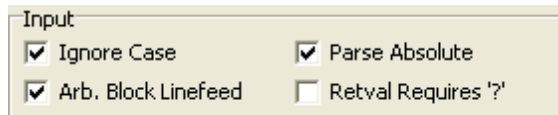
**Command Syntax:** :Preferences:RemoteEnableMacro *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteEnableMacro **False**

**Related Command(s):** RemoteEnableMacro?

**Description:** Sets the enabled status of GPIB macros.



## RemoteArbBlockLF?

**Command Syntax:** :Preferences:RemoteArbBlockLF?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteArbBlockLF] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:RemoteArbBlockLF?  
 [:Preferences:RemoteArbBlockLF] **False**

**Related Command(s):** RemoteArbBlockLF

**Description:** Queries whether a LF (linefeed) character is required to terminate input arbitrary block data.

## RemoteArbBlockLF

**Command Syntax:** :Preferences:RemoteArbBlockLF *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteArbBlockLF False

**Related Command(s):** RemoteArbBlockLF?

**Description:** Sets whether a LF (linefeed) character is required to terminate input arbitrary block data.

## RemoteIgnoreCase?

**Command Syntax:** :Preferences:RemoteIgnoreCase?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteIgnoreCase] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:RemoteIgnoreCase?  
 [:Preferences:RemoteIgnoreCase] False

**Related Command(s):** RemoteIgnoreCase

**Description:** Queries whether case is ingored when parsing received gpib commands.

## RemoteIgnoreCase

**Command Syntax:** :Preferences:RemoteIgnoreCase *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteIgnoreCase False

**Related Command(s):** RemoteIgnoreCase?

**Description:** Sets whether case is ingored when parsing received gpib commands.

## RemoteParseMode?

**Command Syntax:** :Preferences:RemoteParseMode?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteParseMode] *Value*

**Response Argument(s):** *Value* <int> {pmParseRelative=0 | pmParseAbsolute=1}

**Example:** :Preferences:RemoteParseMode?  
 [:Preferences:RemoteParseMode] pmParseRelative

**Related Command(s):** RemoteParseMode

**Description:** Queries whether GPIB object descriptions must include the complete path from the root or are relative to the last referenced object.

## RemoteParseMode

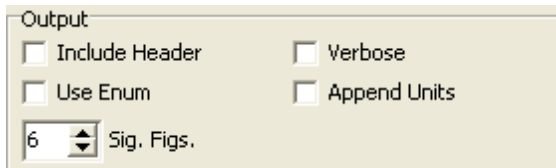
**Command Syntax:** :Preferences:RemoteParseMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {pmParseRelative=0 | pmParseAbsolute=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteParseMode pmParseRelative

**Related Command(s):** RemoteParseMode?

**Description:** Sets whether GPIB object descriptions must include the complete path from the root or are relative to the last referenced object..



## RemoteAppendUnits?

**Command Syntax:** :Preferences:RemoteAppendUnits?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteAppendUnits] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:RemoteAppendUnits?

**[ :Preferences:RemoteAppendUnits ] False**

**Related Command(s):** RemoteAppendUnits

**Description:** Queries whether unit strings will be appended to the responses to GPIB queries.

## RemoteAppendUnits

**Command Syntax:** :Preferences:RemoteAppendUnits *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteAppendUnits False

**Related Command(s):** RemoteAppendUnits?

**Description:** Sets whether unit strings will be appended to the responses to GPIB queries.

## RemoteSigFig?

**Command Syntax:** :Preferences:RemoteSigFig?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteSigFig] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Preferences:RemoteSigFig?  
[:Preferences:RemoteSigFig] **Value**

**Related Command(s):** RemoteSigFig

**Description:** Queries the number of significant figures used in responses to GPIB queries.

## RemoteSigFig

**Command Syntax:** :Preferences:RemoteSigFig *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteSigFig *Value*

**Related Command(s):** RemoteSigFig?

**Description:** Sets the number of significant figures used in responses to GPIB queries.

## RemoteUseEnum?

**Command Syntax:** :Preferences:RemoteUseEnum?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteUseEnum] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:RemoteUseEnum?  
[:Preferences:RemoteUseEnum] **False**

**Related Command(s):** RemoteUseEnum

**Description:** Queries whether responses will be sent as integers or as ascii enumerations.

## RemoteUseEnum

**Command Syntax:** :Preferences:RemoteUseEnum *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteUseEnum **False**

**Related Command(s):** RemoteUseEnum?

**Description:** Sets whether responses will be sent as integers or as ascii enumerations.

## RemoteUseHeader?

**Command Syntax:** :Preferences:RemoteUseHeader?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteUseHeader] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Preferences:RemoteUseHeader?

**[ :Preferences:RemoteUseHeader ] False**

**Related Command(s):** RemoteUseHeader

**Description:** Queries whether responses will include the command header.

## RemoteUseHeader

**Command Syntax:** :Preferences:RemoteUseHeader Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteUseHeader False

**Related Command(s):** RemoteUseHeader?

**Description:** Sets whether responses will include the command header.

## RemoteVerbose?

**Command Syntax:** :Preferences:RemoteVerbose?

**Command Argument(s):**

**Response Syntax:** [:Preferences:RemoteVerbose] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Preferences:RemoteVerbose?

**[ :Preferences:RemoteVerbose ] False**

**Related Command(s):** RemoteVerbose

**Description:** Queries the on/off status of verbose mode.

## RemoteVerbose

**Command Syntax:** :Preferences:RemoteVerbose Value [, AllowCoercion]

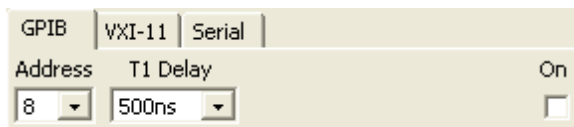
**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:RemoteVerbose False

**Related Command(s):** RemoteVerbose?

**Description:** Sets the on/off status of verbose mode.



## GPIBAddress?

**Command Syntax:** :Preferences:GPIBAddress?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GPIBAddress] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Preferences:GPIBAddress?  
[:Preferences:GPIBAddress] 8

**Related Command(s):** GPIBAddress

**Description:** Queries the primary GPIB address of the instrument.

## GPIBAddress

**Command Syntax:** :Preferences:GPIBAddress *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:GPIBAddress 8

**Related Command(s):** GPIBAddress?

**Description:** Sets the primary GPIB address of the instrument.

## GPIBEnabled?

**Command Syntax:** :Preferences:GPIBEnabled?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GPIBEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:GPIBEnabled?  
[:Preferences:GPIBEnabled] False

**Related Command(s):** GPIBEnabled

**Description:** Queries whether the GPIB (IEEE-488) interface is enabled.

## GPIBEnabled

**Command Syntax:** :Preferences:GPIBEnabled *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:GPIBEnabled False

**Related Command(s):** GPIBEnabled?

**Description:** Sets whether the GPIB (IEEE-488) interface is enabled.



## GPIBT1Delay?

**Command Syntax:** :Preferences:GPIBT1Delay?

**Command Argument(s):**

**Response Syntax:** [:Preferences:GPIBT1Delay] *Value*

**Response Argument(s):** *Value* <int> {td1100ns=0 | td500ns=1 | td350ns=2}

**Example:** :Preferences:GPIBT1Delay?  
[:Preferences:GPIBT1Delay] td1100ns

**Related Command(s):** GPIBT1Delay

**Description:** Queries the GPIB T1 delay value.

## GPIBT1Delay

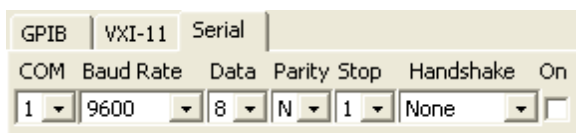
**Command Syntax:** :Preferences:GPIBT1Delay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {td1100ns=0 | td500ns=1 | td350ns=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:GPIBT1Delay td1100ns

**Related Command(s):** GPIBT1Delay?

**Description:** Sets the GPIB T1 delay value.



## SerialBitRate?

**Command Syntax:** :Preferences:SerialBitRate?

**Command Argument(s):**

**Response Syntax:** [:Preferences:SerialBitRate] *Value*

**Response Argument(s):** *Value* <int> {br1200=0 | br1800=1 | br2400=2 | br3600=3 | br4800=4 | br7200=5 | br9600=6 | br14400=7 | br19200=8 | br28800=9 | br38400=10 | br57600=11 | br115200=12}

**Example:** :Preferences:SerialBitRate?  
[:Preferences:SerialBitRate] br1200

**Related Command(s):** SerialBitRate

**Description:** Queries the speed (bit rate) of the rear panel serial interface.

## SerialBitRate

**Command Syntax:** :Preferences:SerialBitRate *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {br1200=0 | br1800=1 | br2400=2 | br3600=3 | br4800=4 | br7200=5 | br9600=6 | br14400=7 | br19200=8 | br28800=9 | br38400=10 | br57600=11 | br115200=12}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:SerialBitRate br1200

**Related Command(s):** SerialBitRate?

**Description:** Sets the speed (bit rate) of the rear panel serial interface.

## SerialDataBits?

**Command Syntax:** :Preferences:SerialDataBits?

**Command Argument(s):**

**Response Syntax:** [:Preferences:SerialDataBits] *Value*

**Response Argument(s):** *Value* <int> {db7=0 | db8=1}

**Example:** :Preferences:SerialDataBits?  
 [:Preferences:SerialDataBits] db7

**Related Command(s):** SerialDataBits

**Description:** Queries the number of data bits for the serial interface.

## SerialDataBits

**Command Syntax:** :Preferences:SerialDataBits *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {db7=0 | db8=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:SerialDataBits db7

**Related Command(s):** SerialDataBits?

**Description:** Sets the number of data bits for the serial interface.

## SerialEnabled?

**Command Syntax:** :Preferences:SerialEnabled?

**Command Argument(s):**

**Response Syntax:** [:Preferences:SerialEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:SerialEnabled?  
 [:Preferences:SerialEnabled] False

**Related Command(s):** SerialEnabled

**Description:** Queries the enabled status of the serial interface.

## SerialEnabled

**Command Syntax:** `:Preferences:SerialEnabled Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:SerialEnabled False`

**Related Command(s):** SerialEnabled?

**Description:** Sets the enabled status of the serial interface.

## SerialHandshake?

**Command Syntax:** `:Preferences:SerialHandshake?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:SerialHandshake] Value`

**Response Argument(s):** `Value <int> {hsNone=0 | hsRTSCTS=1 | hsDTRDSR=2 | hsXonXoff=3}`

**Example:** `:Preferences:SerialHandshake?`

`[:Preferences:SerialHandshake] hsNone`

**Related Command(s):** SerialHandshake

**Description:** Queries the serial interface handshake mode.

## SerialHandshake

**Command Syntax:** `:Preferences:SerialHandshake Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {hsNone=0 | hsRTSCTS=1 | hsDTRDSR=2 | hsXonXoff=3}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Preferences:SerialHandshake hsNone`

**Related Command(s):** SerialHandshake?

**Description:** Sets the serial interface handshake mode.

## SerialParity?

**Command Syntax:** `:Preferences:SerialParity?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:SerialParity] Value`

**Response Argument(s):** `Value <int> {ptNone=0 | ptOdd=1 | ptEven=2}`

**Example:** `:Preferences:SerialParity?`

`[:Preferences:SerialParity] ptNone`

**Related Command(s):** SerialParity

**Description:** Queries the serial interface parity selection.

## SerialParity

**Command Syntax:** :Preferences:SerialParity *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ptNone=0 | ptOdd=1 | ptEven=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:SerialParity ptNone

**Related Command(s):** SerialParity?

**Description:** Sets the serial interface parity selection.

## SerialPort?

**Command Syntax:** :Preferences:SerialPort?

**Command Argument(s):**

**Response Syntax:** [:Preferences:SerialPort] *Value*

**Response Argument(s):** *Value* <int> {peCOM1=0 | peCOM2=1 | peCOM3=2 | peCOM4=3}

**Example:** :Preferences:SerialPort?  
 [:Preferences:SerialPort] peCOM1

**Related Command(s):** SerialPort

**Description:** Queries the COM port used by the serial interface. The COM port is only changeable when SR1 is run in demo mode. On the instrument the COM port is fixed.

## SerialPort

**Command Syntax:** :Preferences:SerialPort *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {peCOM1=0 | peCOM2=1 | peCOM3=2 | peCOM4=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:SerialPort peCOM1

**Related Command(s):** SerialPort?

**Description:** Sets the COM port used by the serial interface. The COM port is only changeable when SR1 is run in demo mode. On the instrument the COM port is fixed.

## SerialStopBits?

**Command Syntax:** :Preferences:SerialStopBits?

**Command Argument(s):**

**Response Syntax:** [:Preferences:SerialStopBits] *Value*

**Response Argument(s):** *Value* <int> {sb1=0 | sb2=1}

**Example:** :Preferences:SerialStopBits?  
 [:Preferences:SerialStopBits] sb1

**Related Command(s):** SerialStopBits

**Description:** Queries the number of stop bits for the serial interface.

## SerialStopBits

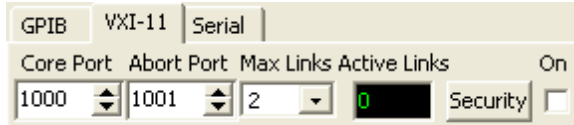
**Command Syntax:** `:Preferences:SerialStopBits Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {sb1=0|sb2=1}`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Preferences:SerialStopBits sb1`

**Related Command(s):** SerialStopBits?

**Description:** Sets the number of stop bits for the serial interface



## VXIAbortPort?

**Command Syntax:** `:Preferences:VXIAbortPort?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:VXIAbortPort] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Preferences:VXIAbortPort?`  
`[:Preferences:VXIAbortPort] 1001`

**Related Command(s):** VXIAbortPort

**Description:** Queries the Abort Port number for the VXI11 interface.

## VXIAbortPort

**Command Syntax:** `:Preferences:VXIAbortPort Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`  
`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:Preferences:VXIAbortPort 1001`

**Related Command(s):** VXIAbortPort?

**Description:** Sets the Abort Port number for the VXI11 interface.

## VXIActiveLinksRdg?

**Command Syntax:** `:Preferences:VXIActiveLinksRdg?`

**Command Argument(s):**

**Response Syntax:** `[:Preferences:VXIActiveLinksRdg] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:Preferences:VXIActiveLinksRdg?`  
`[:Preferences:VXIActiveLinksRdg] 2`

**Description:** Queries the number of sessions currently active on the VXI-11 interface.

## VXIAllowList?

**Command Syntax:** :Preferences:VXIAllowList?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIAllowList] *Value*

**Response Argument(s):** *Value* <string>

**Example:** :Preferences:VXIAllowList?

**[ :Preferences:VXIAllowList ] "\*"\*.\*.\*.\*,192.168.0.1"**

**Related Command(s):** VXIAllowList

**Description:** Returns the list of "allowed" IP addresses that can connect via VXI-11. Each line in the list is separated by a comma.

## VXIAllowList

**Command Syntax:** :Preferences:VXIAllowList *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <string>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:VXIAllowList "192.168.\*.\*,10.0.1.1"

**Related Command(s):** VXIAllowList?

**Description:** Sets the list of allowed IP addresses. The input argument is a string of comma separated IP addresses. The "\*" wildcard character can be used in the list.

## VXICheckDeniedFirst?

**Command Syntax:** :Preferences:VXICheckDeniedFirst?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXICheckDeniedFirst] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:VXICheckDeniedFirst?

**[ :Preferences:VXICheckDeniedFirst ] False**

**Related Command(s):** VXICheckDeniedFirst

**Description:** Queries whether the denied list is checked before the allowed list when determining whether an IP address is allowed to connect.

## VXICheckDeniedFirst

**Command Syntax:** :Preferences:VXICheckDeniedFirst *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:VXICheckDeniedFirst False

**Related Command(s):** VXICheckDeniedFirst?

**Description:** Sets whether the denied list is checked before the allowed list when determining whether an IP address is allowed to connect.

## VXICorePort?

**Command Syntax:** :Preferences:VXICorePort?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXICorePort] Value

**Response Argument(s):** Value <int>

**Example:** :Preferences:VXICorePort?  
[:Preferences:VXICorePort] 1000

**Related Command(s):** VXICorePort

**Description:** Queries the value of the core VXI-11 port.

## VXICorePort

**Command Syntax:** :Preferences:VXICorePort Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Preferences:VXICorePort 1000

**Related Command(s):** VXICorePort?

**Description:** Sets the value of the core VXI-11 port.

## VXIDenyList?

**Command Syntax:** :Preferences:VXIDenyList?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIDenyList] Value

**Response Argument(s):** Value <string>

**Example:** :Preferences:VXIDenyList?  
[:Preferences:VXIDenyList] "75.101.142.50"

**Related Command(s):** VXIDenyList

**Description:** Queries the list of denied IP addresses. Value is returned as a comma separated string.

## VXIDenyList

**Command Syntax:** :Preferences:VXIDenyList Value [, AllowCoercion]

**Command Argument(s):** Value <string>  
AllowCoercion <bool> {False=0|True=1}

**Example:** :Preferences:VXIDenyList "75.101.142.50"

**Related Command(s):** VXIDenyList?

**Description:** Sets the list of denied IP addresses. The input argument is a comma separated list of denied IP addresses.

## VXIEnabled?

**Command Syntax:** :Preferences:VXIEnabled?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Preferences:VXIEnabled?  
[:Preferences:VXIEnabled] False

**Related Command(s):** VXIEnabled

**Description:** Queries the enabled status of the VXI-11 interface.

## VXIEnabled

**Command Syntax:** :Preferences:VXIEnabled *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:VXIEnabled False

**Related Command(s):** VXIEnabled?

**Description:** Sets the enabled status of the VXI-11 interface.

## VXIMaxLinks?

**Command Syntax:** :Preferences:VXIMaxLinks?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIMaxLinks] *Value*

**Response Argument(s):** *Value* <int> {ml1=0 | ml2=1 | ml5=2 | ml10=3 | mlMax=4}

**Example:** :Preferences:VXIMaxLinks?  
[:Preferences:VXIMaxLinks] ml1

**Related Command(s):** VXIMaxLinks

**Description:** Queries the maximum number of sessions that can simultaneously be supported by the VXI-11 interface.

## VXIMaxLinks

**Command Syntax:** :Preferences:VXIMaxLinks *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ml1=0 | ml2=1 | ml5=2 | ml10=3 | mlMax=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Preferences:VXIMaxLinks ml1

**Related Command(s):** VXIMaxLinks?

**Description:** Sets the maximum number of sessions that can simultaneously be supported by the VXI-11 interface.



## VXIPassword?

**Command Syntax:** :Preferences:VXIPassword?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIPassword] Value

**Response Argument(s):** Value <string>

**Example:** :Preferences:VXIPassword?  
[:Preferences:VXIPassword] Value

**Related Command(s):** VXIPassword

**Description:** Queries the VXI-11 connection password.

## VXIPassword

**Command Syntax:** :Preferences:VXIPassword Value [, AllowCoercion]

**Command Argument(s):** Value <string>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:VXIPassword Value

**Related Command(s):** VXIPassword?

**Description:** Sets the VXI-11 connection password.

## VXIRequirePassword?

**Command Syntax:** :Preferences:VXIRequirePassword?

**Command Argument(s):**

**Response Syntax:** [:Preferences:VXIRequirePassword] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Preferences:VXIRequirePassword?  
[:Preferences:VXIRequirePassword] False

**Related Command(s):** VXIRequirePassword

**Description:** Queries whether a password is required to connect to the VXI-11 interface.

## VXIRequirePassword

**Command Syntax:** :Preferences:VXIRequirePassword Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Preferences:VXIRequirePassword False

**Related Command(s):** VXIRequirePassword?

**Description:** Sets whether a password is required to connect to the VXI-11 interface.

### *Supported Form Commands:*

**:Preferences:OpenForm**  
**:Preferences:OpenFormwID?**  
**:Preferences:CloseForm**  
**:Preferences:CloseForms**  
**:Preferences:FormCount?**  
**:Preferences:FormID?**

### 2.3.15 Scripting

<b>Object:</b>	<b>:Scripting :Script</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the local scripting environment of SR1.

#### FileName?

*Command Syntax:* :Scripting:FileName?

*Command Argument(s):*

*Response Syntax:* [:Scripting:FileName] Value

*Response Argument(s):* Value <string>

*Example:* :Scripting:FileName?

**[:Scripting:FileName] "MyScript.vbs"**

*Description:* Queries the filename of the current script file.

#### IsModified?

*Command Syntax:* :Scripting:IsModified?

*Command Argument(s):*

*Response Syntax:* [:Scripting:IsModified] Value

*Response Argument(s):* Value <int> {False=0 | True=1}

*Example:* :Scripting:IsModified?

**[:Scripting:IsModified] False**

*Description:* Queries whether the script has been modified since the last save.

#### Language?

*Command Syntax:* :Scripting:Language?

*Command Argument(s):*

*Response Syntax:* [:Scripting:Language] Value

*Response Argument(s):* Value <int> {scVBScript=0 | scJScript=1 | scPython=2}

*Example:* :Scripting:Language?

**[:Scripting:Language] scVBScript**

*Related Command(s):* Language

*Description:* Queries the script language selection.

## Language

**Command Syntax:** `:Scripting:Language Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {scVBScript=0 | scJScript=1 | scPython=2}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Scripting:Language scVBScript`

**Related Command(s):** Language?

**Description:** Sets the script language selection.

## ShowUI?

**Command Syntax:** `:Scripting:ShowUI?`

**Command Argument(s):**

**Response Syntax:** `[:Scripting:ShowUI] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Scripting:ShowUI?`

`[:Scripting:ShowUI] False`

**Related Command(s):** ShowUI

**Description:** Queries whether the right-hand panels of the scripting window display the SR1 Basic user interface.

## ShowUI

**Command Syntax:** `:Scripting:ShowUI Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Scripting:ShowUI False`

**Related Command(s):** ShowUI?

**Description:** Sets whether the right-hand panels of the scripting window display the SR1 Basic user interface.

## Terminate?

**Command Syntax:** `:Scripting:Terminate?`

**Command Argument(s):**

**Response Syntax:** `[:Scripting:Terminate] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Scripting:Terminate?`

`[:Scripting:Terminate] False`

**Related Command(s):** Terminate

**Description:** Queries the value of the script terminate flag. The script terminate flag can be checked by scripts as a signal to terminate processing.

## Terminate

**Command Syntax:** `:Scripting:Terminate Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Scripting:Terminate False`

**Related Command(s):** Terminate?

**Description:** Sets the value of the script terminate flag. The script terminate flag can be checked by scripts as a signal to terminate processing.

## ClearLines

**Command Syntax:** `:Scripting:ClearLines`

**Command Argument(s):** None

**Example:** `:Scripting:ClearLines`

**Description:** Clears all lines of the current script.

## IncludeScript

**Command Syntax:** `:Scripting:IncludeScript FileName, Location`

**Command Argument(s):** `FileName <string>`  
`Location <int> {ilUserScript=0 | ilEventScript=1}`

**Example:** `:Scripting:IncludeScript "MyScript.vbs", ilUserScript`

**Description:** This command tells SR1 to include other script files when executing the current script. This enables the user to use subroutines defined in other files. The Location variable informs SR1 to include the script file in the User script processor, or the Events script processor.

## Load

**Command Syntax:** `:Scripting:Load FileName`

**Command Argument(s):** `FileName <string>`

**Example:** `:Scripting:Load "MyScript.vbs"`

**Description:** Loads the script with the specified filename.

## New

**Command Syntax:** `:Scripting:New`

**Command Argument(s):** None

**Example:** `:Scripting:New`

**Description:** Clears the current script and creates a new, empty script with the filename "newscrip.vbs".

## Run

*Command Syntax:* :Scripting:Run *Timeout*

*Command Argument(s):* *Timeout* <int>

*Example:* :Scripting:Run Value

*Description:* Runs the current script.

## Save

*Command Syntax:* :Scripting:Save

*Command Argument(s):* None

*Example:* :Scripting:Save

*Description:* Saves the current script to the current scripting file.

## SaveAs

*Command Syntax:* :Scripting:SaveAs *FileName*

*Command Argument(s):* *FileName* <string>

*Example:* :Scripting:SaveAs Value

*Description:* Saves the current script to a new scripting file and make that file the current file.

## StopEvents

*Command Syntax:* :Scripting:StopEvents

*Command Argument(s):* None

*Example:* :Scripting:StopEvents

*Description:* Stops script processing of events.

## Scripting Window Form Commands

### OpenForm

*Command Syntax:* :Scripting:OpenForm

*Command Argument(s):* None

*Example:* :Scripting:OpenForm

*Description:* Opens a scripting window on the current page of the page control.

## CloseForm

*Command Syntax:* :Scripting:CloseForm

*Command Argument(s):* None

*Example:* :Scripting:CloseForm

*Description:* Closes the scripting window.

## FormID?

*Command Syntax:* :Scripting:FormID?

*Command Argument(s):* None

*Response Syntax:* [:Scripting:FormID] *FormID*

*Response Argument(s):* *FormID* <int>

*Example:* :Scripting:FormID?

**[:Scripting:FormID] 5**

*Description:* Returns the FormID of the scripting window.

## Scripting Log Commands

### OpenScriptLogForm

*Command Syntax:* :Scripting:OpenScriptLogForm

*Command Argument(s):* None

*Example:* :Scripting:OpenScriptLogForm

*Description:* Opens a script log form on the current page of the page control.

### OpenScriptLogFormwID?

*Command Syntax:* :Scripting:OpenScriptLogFormwID?

*Command Argument(s):* None

*Response Syntax:* [:Scripting:OpenScriptLogFormwID] *FormID*

*Response Argument(s):* *FormID* <int>

*Example:* :Scripting:OpenScriptLogFormwID?

**[:Scripting:OpenScriptLogFormwID] 21**

*Description:* Opens a script log form on the current page of the page control and returns the FormID of the newly created form.

## ScriptLogFormCount?

*Command Syntax:* :Scripting:ScriptLogFormCount?

*Command Argument(s):* None

*Response Syntax:* [:Scripting:ScriptLogFormCount] Count

*Response Argument(s):* Count <int>

*Example:* :Scripting:ScriptLogFormCount?  
[:Scripting:ScriptLogFormCount] 1

*Description:* Counts the number of script log forms on all pages of the page control.

## ScriptLogFormID?

*Command Syntax:* :Scripting:ScriptLogFormID? Index

*Command Argument(s):* Index <int>

*Response Syntax:* [:Scripting:ScriptLogFormID] FormID

*Response Argument(s):* FormID <int>

*Example:* :Scripting:ScriptLogFormID? Value  
[:Scripting:ScriptLogFormID] 5

*Description:* Returns the FormID of the Index<sup>th</sup> script log form.

## CloseScriptLogForm

*Command Syntax:* :Scripting:CloseScriptLogForm FormID

*Command Argument(s):* FormID <int>

*Example:* :Scripting:CloseScriptLogForm 5

*Description:* Closes the script log form with the given FormID.

## CloseScriptLogForms

*Command Syntax:* :Scripting:CloseScriptLogForms

*Command Argument(s):* None

*Example:* :Scripting:CloseScriptLogForms

*Description:* Closes all script log forms on all pages of the page control.

## WriteLine

*Command Syntax:* :Scripting:WriteLine Text

*Command Argument(s):* Text <string>

*Example:* :Scripting:WriteLine "Output=100"

*Description:* Writes a new line to the script log form.



## PrintLog

*Command Syntax:* :Scripting:PrintLog

*Command Argument(s):* None

*Example:* :Scripting:PrintLog

*Description:* Prints the contents of the script log form to the currently configured SR1 printer.

### 2.3.16 Displays

<b>Object:</b>	<b>:Displays</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to SR1's collection of displays

#### Load

**Command Syntax:** :Displays:Load *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Displays:Load "DisplayFile.XML"

**Description:** Loads the indicated display file.

#### SaveAll

**Command Syntax:** :Displays:SaveAll *FileName*, *GraphData*

**Command Argument(s):** *FileName* <string>

*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Displays:SaveAll "MyFile.XML", gdNeverSave

**Description:** Saves all displays on all pages of the page control to the indicated file. The graph data option indicates whether graph data should always be saved in the file, never saved, or saved only for offline traces.

#### DeleteAll

**Command Syntax:** :Displays>DeleteAll

**Command Argument(s):** None

**Example:** :Displays>DeleteAll

**Description:** Deletes all displays on all pages of the page control.

## Bar Chart Commands

#### NewBar?

**Command Syntax:** :Displays:NewBar?

**Command Argument(s):** None

**Response Syntax:** [:Displays:NewBar] *BarID*

**Response Argument(s):** *BarID* <int>

**Example:** :Displays:NewBar?

**[ :Displays:NewBar ] 5**

**Description:** Creates a new bar chart on the current page of the page control and returns the objectID of the newly created bar chart. The properties of the chart can subsequently be manipulated using the object :Displays:Bar(*BarID*).

## BarCount?

**Command Syntax:** :Displays:BarCount?

**Command Argument(s):** None

**Response Syntax:** [:Displays:BarCount] Count

**Response Argument(s):** Count <int>

**Example:** :Displays:BarCount?  
[:Displays:BarCount] 2

**Description:** Returns the total number of bar charts found on all pages of the page control.

## BarItem?

**Command Syntax:** :Displays:BarItem? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:Displays:BarItem] Bar

**Response Argument(s):** Bar <itemstring>

**Example:** :Displays:BarItem? 0  
[:Displays:BarItem] 5

**Description:** Returns the objectID of the Index<sup>th</sup> bar chart. The properties of the chart can subsequently be manipulated using the object Displays:Bar(objectID).

## Digitizer Display Commands

### NewDigitizerDisplay?

**Command Syntax:** :Displays:NewDigitizerDisplay?

**Command Argument(s):** None

**Response Syntax:** [:Displays:NewDigitizerDisplay] DigitizerID

**Response Argument(s):** DigitizerID <int>

**Example:** :Displays:NewDigitizerDisplay?  
[:Displays:NewDigitizerDisplay] 7

**Description:** Creates a new digitizer display on the current page of the page control and returns the objectID of the newly created digitizer display. The properties of the chart can subsequently be manipulated using the object :Displays:DigitizerDisplay (digitizerID).

### DigitizerDisplayCount?

**Command Syntax:** :Displays:DigitizerDisplayCount?

**Command Argument(s):** None

**Response Syntax:** [:Displays:DigitizerDisplayCount] Count

**Response Argument(s):** Count <int>

**Example:** :Displays:DigitizerDisplayCount?  
[:Displays:DigitizerDisplayCount] 1

**Description:** Returns the total number of digitizer displays on all pages of the page control.

## DigitizerDisplayItem?

**Command Syntax:** :Displays:DigitizerDisplayItem? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:Displays:DigitizerDisplayItem] *DigitizerDisplay*

**Response Argument(s):** *DigitizerDisplay* <itemstring>

**Example:** :Displays:DigitizerDisplayItem? 0  
[:Displays:DigitizerDisplayItem] 10

**Description:** Returns the objectID of the Index<sup>th</sup> digitizer display. The properties of the display can subsequently be manipulated using the object Displays:DigitizerDisplay (objectID).

## Graph Commands

### NewGraph?

**Command Syntax:** :Displays:NewGraph?

**Command Argument(s):** None

**Response Syntax:** [:Displays:NewGraph] *GraphID*

**Response Argument(s):** *GraphID* <int>

**Example:** :Displays:NewGraph?  
[:Displays:NewGraph] 5

**Description:** Creates a new graph on the current page of the page control and returns the id of the newly created graph. The properties of the graph can subsequently be manipulated using the object :Displays:Graph(id).

### GraphCount?

**Command Syntax:** :Displays:GraphCount?

**Command Argument(s):** None

**Response Syntax:** [:Displays:GraphCount] *Count*

**Response Argument(s):** *Count* <int>

**Example:** :Displays:GraphCount?  
[:Displays:GraphCount] 4

**Description:** Returns the total number of graphs on all pages of the page control.

## GraphItem?

**Command Syntax:** :Displays:GraphItem? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:Displays:GraphItem] *Graph*

**Response Argument(s):** *Graph* <itemstring>

**Example:** :Displays:GraphItem? 0  
[:Displays:GraphItem] 21

**Description:** Returns the objectID of the Index<sup>th</sup> graph. The properties of the graph can subsequently be manipulated using the object Displays:Graph(objectID).

## 2.3.16.1 Graph

<b>Object:</b>	<b>:Displays:Graph(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the graph with objectID <i>i</i> .

**Close**

**Command Syntax:** :Displays:Graph(*i*):Close

**Command Argument(s):** None

**Example:** :Displays:Graph(1):Close

**Description:** Closes the graph.

**BackgroundColor?**

**Command Syntax:** :Displays:Graph(*i*):BackgroundColor?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(*i*):BackgroundColor] *Value*

**Response Argument(s):** *Value* <int> {bcWhite=0 | bcBlack=1}

**Example:** :Displays:Graph(1):BackgroundColor?

**[:Displays:Graph(1):BackgroundColor] bcWhite**

**Related Command(s):** BackgroundColor

**Description:** Queries whether the graph has a white or black background.

**BackgroundColor**

**Command Syntax:** :Displays:Graph(*i*):BackgroundColor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {bcWhite=0 | bcBlack=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):BackgroundColor bcWhite

**Related Command(s):** BackgroundColor?

**Description:** Sets the graph background to white or black.

## Title?

**Command Syntax:** :Displays:Graph(i):Title?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Title] Value

**Response Argument(s):** Value <string>

**Example:** :Displays:Graph(1):Title?  
[:Displays:Graph(1):Title] Graph 0

**Related Command(s):** Title

**Description:** Queries the graph title.

## Title

**Command Syntax:** :Displays:Graph(i):Title Value [, AllowCoercion]

**Command Argument(s):** Value <string>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Title Value

**Related Command(s):** Title?

**Description:** Sets the graph title.

## FormID?

**Command Syntax:** :Displays:Graph(i):FormID?

**Command Argument(s):** None

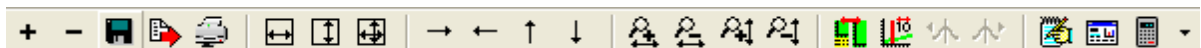
**Response Syntax:** [:Displays:Graph(i):FormID] FormID

**Response Argument(s):** FormID <int>

**Example:** :Displays:Graph(1):FormID?  
[:Displays:Graph(1):FormID] Value

**Description:** Queries the FormID for the graph. The FormID can be used as an argument to the standard form [commands](#) for setting and getting the form's positions, size, etc.

## Toolbar Commands



## AddEQTrace?

**Command Syntax:** :Displays:Graph(i):AddEQTrace? *FileName*, *Type*

**Command Argument(s):** *FileName* <string>  
*Type* <int> {eqAmp=0|eqAmpInv=1|eqPhase=2}

**Response Syntax:** [:Displays:Graph(i):AddEQTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):AddEQTrace? "Aweight.eq", eqAmp  
[:Displays:Graph(1):AddEQTrace] 20

**Description:** Adds a trace to the graph corresponding to the specified EQ file. The data in the trace will either be the EQ file amplitude, the inverse of the EQ file amplitude, or the EQ file phase depending on the value of the "type" argument. The command returns the id of the newly created trace.

## AddTrace?

**Command Syntax:** :Displays:Graph(i):AddTrace? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0|msAnlgFreqA=1|msAnlgFreqB=2|msAnlgPhase=3|msDigFreqA=10|msDigFreqB=11|msDigPhase=12|msDigCarrierFreq=14|msDigCarrierAmp=15|msDigCarrierDelay=16|msA0LevelA=1100|msA0LevelB=1101|msA0FFTtimeRec=1110|msA0FFT2spectrum=1111|msA0FFT2linSpec=1112|msA0FFT2linPhase=1113|msA0FFT2timeRecA=1120|msA0FFT2timeRecB=1121|msA0FFT2spectrumA=1122|msA0FFT2spectrumB=1123|msA0FFT2linSpecA=1124|msA0FFT2linSpecB=1125|msA0FFT2xferMag=1126|msA0FFT2xferPhase=1127|msA0FFT2coherence=1128|msA0FFT2impulseResp=1129|msA0TimeDomDetector=1130|msA0FFT2anechoicRespMag=1131|msA0FFT2anechoicRespPhase=1132|msA0FFT2energyTimeCurve=1133|msA0THD0=1150|msA0THD1=1151|msA0THDvector=1152|msA0Imd=1160|msA0jitFreqDomTimeRec=1170|msA0jitFreqDomPower=1171|msA0jitFreqDomLinSpec=1172|msA0jitFreqDomLinPhase=1173|msA0jitFreqDomJitter=1174|msA0jitTimeDomJitter=1175|msA0jitPhysSampRate=1176|msA0HistoTimeRecA=1180|msA0HistoTimeRecB=1181|msA0HistoHistoA=1182|msA0HistoHistoB=1183|msA0HistoProbA=1184|msA0HistoProbB=1185|msA0HistoFitA=1186|msA0HistoFitB=1187|msA0HistoFitMeanA=1188|msA0HistoFitMeanB=1189|msA0HistoFitSigmaA=1190|msA0HistoFitSigmaB=1191|msA0MTtimeRecA=1200|msA0MTtimeRecB=1201|msA0MTspectrumA=1202|msA0MTspectrumB=1203|msA0MTthdnBinsA=1204|msA0MTthdnBinsB=1205|msA0MTthdBinsA=1206|msA0MTthdBinsB=1207|msA0MTimdBinsA=1208|msA0MTimdBinsB=1209|msA0MTnoiseBinsA=1210|msA0MTnoiseBinsB=1211|msA0MTtdBinsA=1212|msA0MTtdBinsB=1213|msA0MTfreqRespMagA=1220|msA0MTfreqRespMagB=1221|msA0MTfreqRespPhaseA=1222|msA0MTfreqRespPhaseB=1223|msA0MTthdnVsFreqA=1224|msA0MTthdnVsFreqB=1225|msA0MTthdVsFreqA=1226|msA0MTthdVsFreqB=1227|msA0MTimdVsFreqA=1228|msA0MTimdVsFreqB=1229|msA0MTxtalkVsFreqAB=1230|msA0MTxtalkVsFreqBA=1231|msA0MTthdnA=1240|msA0MTthdnB=1241|msA0MTthdA=1242|msA0MTthdB=1243|msA0MTimdA=1244|msA0MTimdB=1245|msA0MTnoiseA=1246|msA0MTnoiseB=1247|msA0MTtdA=1248|msA0MTtdB=1249|msA0MTrippleA=1250|msA0MTrippleB=1251|msA0MTlowestToneA=1252|msA0MTlowestToneB=1253|msA0MThighestToneA=1254|msA0MThighestToneB=1255|msA1LevelA=2100|



msA1LevelB=2101 | msA1FFFTtimeRec=2110 | msA1FFFTspectrum=2111 |  
msA1FFFTlinSpec=2112 | msA1FFFTlinPhase=2113 | msA1FFFT2timeRecA=2120 |  
msA1FFFT2timeRecB=2121 | msA1FFFT2spectrumA=2122 |  
msA1FFFT2spectrumB=2123 | msA1FFFT2linSpecA=2124 | msA1FFFT2linSpecB=2125 |  
msA1FFFT2xferMag=2126 | msA1FFFT2xferPhase=2127 | msA1FFFT2coherence=2128 |  
msA1FFFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
msA1FFFT2anechoicRespMag=2131 | msA1FFFT2anechoicRespPhase=2132 |  
msA1FFFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |  
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |  
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |  
msA1MTxtalkVsFreqA=2230 | msA1MTxtalkVsFreqB=2231 |  
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |  
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |  
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |  
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |  
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |  
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |  
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |  
msSweep5=5005}

**Response Syntax:** [:Displays:Graph(i):AddTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):AddTrace? msA0FFFTspectrum  
[:Displays:Graph(1):AddTrace] 2

**Description:** Adds a trace to the graph corresponding to the specified measurement and returns the ID of the newly created trace.

## DeleteAllTraces

**Command Syntax:** :Displays:Graph(i):DeleteAllTraces

**Command Argument(s):** None

**Example:** :Displays:Graph(1):DeleteAllTraces

**Description:** Deletes all traces of the graph.

## DeleteSweepTraces

**Command Syntax:** :Displays:Graph(i):DeleteSweepTraces *TraceID*

**Command Argument(s):** *TraceID* <int>

**Example:** :Displays:Graph(1):DeleteSweepTraces Value

**Description:** Deletes all sweep traces.

## DeleteTrace

**Command Syntax:** :Displays:Graph(i):DeleteTrace *TraceID*

**Command Argument(s):** *TraceID* <int>

**Example:** :Displays:Graph(1):DeleteTrace Value

**Description:** Deletes the trace with the specific ID.

## DeleteUnusedTraces

**Command Syntax:** :Displays:Graph(i):DeleteUnusedTraces

**Command Argument(s):** None

**Example:** :Displays:Graph(1):DeleteUnusedTraces

**Description:** Deletes traces with no data.

## Save

**Command Syntax:** :Displays:Graph(i):Save *FileName*, *GraphData*

**Command Argument(s):** *FileName* <string>

*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Displays:Graph(1):Save "MyGraphFile.XML", gdNeverSave

**Description:** Saves the graph to the specified file. If the *GraphData* argument is *gdNeverSave* then no data is saved with the graph. If *gdAlwaysSave* is sent then the graph file will include data for all traces. If *gdSaveOfflineOnly* is specified then data will be saved only for offline traces.

## ExportData

**Command Syntax:** :Displays:Graph(i):ExportData *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Displays:Graph(1):ExportData "MyData.txt"

**Description:** Exports the graph data to a text file with the given file name.

## ExportGraph

**Command Syntax:** :Displays:Graph(i):ExportGraph *FileType*, *FileName*

**Command Argument(s):** *FileType* <int> {ftBitMap=0 | ftEnMetaFile=1 | ftJpeg=2}  
*FileName* <string>

**Example:** :Displays:Graph(1):ExportGraph 0, "GraphFile.BMP"

**Description:** Exports the graph to the specified graphics file type and filename.

## Print

**Command Syntax:** :Displays:Graph(i):Print

**Command Argument(s):** None

**Example:** :Displays:Graph(1):Print

**Description:** Prints the graph to the currently configured default printer.

## AutoScale

**Command Syntax:** :Displays:Graph(i):AutoScale

**Command Argument(s):** None

**Example:** :Displays:Graph(1):AutoScale

**Description:** Autoscales the X and Y axes of the active trace to fit the data.

## AutoScaleX

**Command Syntax:** :Displays:Graph(i):AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:Graph(1):AutoScaleX

**Description:** Autoscales the X-axis of the active trace to fit the data.

## AutoScaleY

**Command Syntax:** :Displays:Graph(i):AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:Graph(1):AutoScaleY

**Description:** Autoscales the Y-axis of the active trace to fit the data.

## ShiftX

**Command Syntax:** :Displays:Graph(i):ShiftX *Direction*

**Command Argument(s):** *Direction* <bool> {xLeft=0 | xRight=1}

**Example:** :Displays:Graph(1):ShiftX xLeft

**Description:** Shifts the data (or axes) of the active trace left or right. Equivalent to the "left/right arrow" buttons on the graph toolbar. Selection of the data or axes (which reverses the left/right selection) is done with the [:Preference:ShiftMode](#) command.

## ShiftY

**Command Syntax:** :Displays:Graph(i):ShiftY *Direction*

**Command Argument(s):** *Direction* <bool> {yDown=0 | yUp=1}

**Example:** :Displays:Graph(1):ShiftY yDown

**Description:** Shifts the data (or axes) of the active trace up or down. Equivalent to the "up/down arrow" buttons on the graph toolbar. Selection of the data or axes (which reverses the up/down selection) is done with the [:Preference:ShiftMode](#) command.

## ZoomX

**Command Syntax:** :Displays:Graph(i):ZoomX *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0 | zmIn=1}

**Example:** :Displays:Graph(1):ZoomX zmOut

**Description:** Zooms the X-axis of the active trace in or out around the current zoom point. If cursors are displayed the zoom point is the active cursor location. If cursors are off the zoom point is the center of the graph. The sense of zooming is affected by the [:Preferences:ZoomMode](#) command.

## ZoomY

**Command Syntax:** :Displays:Graph(i):ZoomY *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0 | zmIn=1}

**Example:** :Displays:Graph(1):ZoomY zmOut

**Description:** Zooms the Y-axis of the active trace in or out around the current zoom point. If cursors are displayed the zoom point is the active cursor location. If cursors are off the zoom point is the center of the graph. The sense of zooming is affected by the [:Preferences:ZoomMode](#) command.

## Maximize

**Command Syntax:** :Displays:Graph(i):Maximize *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Maximize False

**Related Command(s):** Maximize?

**Description:** Queries the "maximized" status of the graph. When the graph is maximized the central graph section expands to fill the space occupied by the scaling window and the trace listing.

## Maximize?

**Command Syntax:** :Displays:Graph(i):Maximize?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Maximize] *Value*

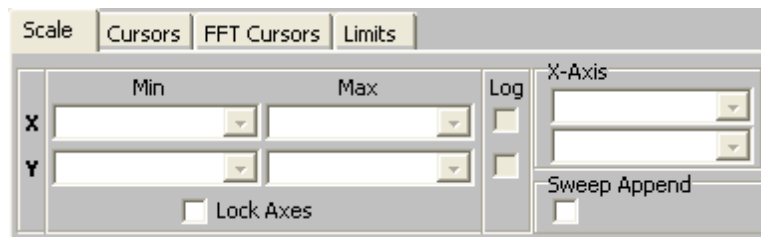
**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Maximize?  
[:Displays:Graph(1):Maximize] False

**Related Command(s):** Maximize

**Description:** Sets the "maximized" status of the graph. When the graph is maximized the central graph section expands to fill the space occupied by the scaling window and the trace listing.

## Scale Commands



These commands, like the controls on the screen, set the properties of either the selected (active) trace or the selected trace and all compatible traces depending on the setting of the "Lock Axes" parameter. To set or query the properties of a specific trace regardless of the state of "Lock Axes" use the :Displays:Graph(i):Trace(j) object.

### xAxis

**Command Syntax:** :Displays:Graph(i):xAxis *Value*

**Command Argument(s):** *Value* <int> {xaIndex=0 | xaTime=1 | xaSweep=2}

**Example:** :Displays:Graph(1):xAxis xaIndex

**Description:** Sets the X-axis selection for scalar traces.

### Xlog

**Command Syntax:** :Displays:Graph(i):Xlog *On*

**Command Argument(s):** *On* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Xlog False

**Description:** Sets logarithmic X-axis on or off.

## Xmax

**Command Syntax:** :Displays:Graph(i):Xmax ValueUnit, Value

**Command Argument(s):** ValueUnit <string>  
Value <double>

**Example:** :Displays:Graph(1):Xmax "HZ", 10000

**Description:** Sets the maximum X-value. Send the empty string "" to set the value in current units.

## Xmin

**Command Syntax:** :Displays:Graph(i):Xmin ValueUnit, Value

**Command Argument(s):** ValueUnit <string>  
Value <double>

**Example:** :Displays:Graph(1):Xmin Value, Value

**Description:** Sets the minimum X-value. Send the empty string "" to set the value in current units.

## xScaleMode

**Command Syntax:** :Displays:Graph(i):xScaleMode Value

**Command Argument(s):** Value <int> {xaFixed=0 | xaPan=1 | xaFull=2}

**Example:** :Displays:Graph(1):xScaleMode xaFixed

**Description:** Sets the scaling mode for scalar traces.

## Ylog

**Command Syntax:** :Displays:Graph(i):Ylog On

**Command Argument(s):** On <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Ylog False

**Description:** Sets logarithmic Y-axis on or off.

## Ymax

**Command Syntax:** :Displays:Graph(i):Ymax ValueUnit, Value

**Command Argument(s):** ValueUnit <string>  
Value <double>

**Example:** :Displays:Graph(1):Ymax "Vp", 100.0

**Description:** Sets the maximum Y value.

## Ymin

**Command Syntax:** `:Displays:Graph(i):Ymin ValueUnit, Value`

**Command Argument(s):** `ValueUnit <string>`  
`Value <double>`

**Example:** `:Displays:Graph(1):Ymin "Vrms", 0.0`

**Description:** Sets the minimum Y value.

## Append?

**Command Syntax:** `:Displays:Graph(i):Append?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:Graph(i):Append] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Append?`

`[:Displays:Graph(1):Append] False`

**Related Command(s):** Append

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## Append

**Command Syntax:** `:Displays:Graph(i):Append Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Append False`

**Related Command(s):** Append?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

## ApplyAll?

**Command Syntax:** `:Displays:Graph(i):ApplyAll?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:Graph(i):ApplyAll] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):ApplyAll?`

`[:Displays:Graph(1):ApplyAll] False`

**Related Command(s):** ApplyAll

**Description:** Queries

## ApplyAll

**Command Syntax:** :Displays:Graph(i):ApplyAll *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):ApplyAll False

**Related Command(s):** ApplyAll?

**Description:** Sets whether scaling commands will affect only the active trace, or the active trace and all compatible traces.

## Trace Commands



## CopyOfflineSweeps

**Command Syntax:** :Displays:Graph(i):CopyOfflineSweeps

**Command Argument(s):** None

**Example:** :Displays:Graph(1):CopyOfflineSweeps

**Description:** Copies any live sweep traces offline.

## CopyOfflineTrace?

**Command Syntax:** :Displays:Graph(i):CopyOfflineTrace? *TraceID*

**Command Argument(s):** *TraceID* <int>

**Response Syntax:** [:Displays:Graph(i):CopyOfflineTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):CopyOfflineTrace? 1  
 [:Displays:Graph(1):CopyOfflineTrace] 2

**Description:** Copies the specified trace to a new offline trace and returns the traceID of the newly created trace.

## CopyTrace

**Command Syntax:** :Displays:Graph(i):CopyTrace *TraceID*

**Command Argument(s):** *TraceID* <int>

**Example:** :Displays:Graph(1):CopyTrace 1

**Description:** Copies the specified trace to the clipboard.



## CutTrace

**Command Syntax:** :Displays:Graph(i):CutTrace *TraceID*

**Command Argument(s):** *TraceID* <int>

**Example:** :Displays:Graph(1):CutTrace 1

**Description:** Copy the specified trace to the clipboard and then delete it.

## GoOffline

**Command Syntax:** :Displays:Graph(i):GoOffline

**Command Argument(s):** None

**Example:** :Displays:Graph(1):GoOffline

**Description:** Makes all traces of the graph offline.

## IsTraceChecked?

**Command Syntax:** :Displays:Graph(i):IsTraceChecked? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:Displays:Graph(i):IsTraceChecked] *Checked*

**Response Argument(s):** *Checked* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):IsTraceChecked? Value  
[:Displays:Graph(1):IsTraceChecked] False

**Description:** Queries whether the Index<sup>th</sup> trace is checked or not.

## TraceCheck

**Command Syntax:** :Displays:Graph(i):TraceCheck *Index, Check*

**Command Argument(s):** *Index* <int>

*Check* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):TraceCheck Value, False

**Description:** Checks (or un-checks) the Index<sup>th</sup> trace.

## LoadTrace?

**Command Syntax:** :Displays:Graph(i):LoadTrace? *FileName*

**Command Argument(s):** *FileName* <string>

**Response Syntax:** [:Displays:Graph(i):LoadTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):LoadTrace? "MyTraceFile.XML"  
[:Displays:Graph(1):LoadTrace] 1

**Description:** Loads the trace stored in the specified file and returns the traceID of the newly created trace.

## PasteTrace?

**Command Syntax:** :Displays:Graph(i):PasteTrace?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):PasteTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):PasteTrace?  
[:Displays:Graph(1):PasteTrace] Value

**Description:** Pastes the trace currently stored in the clipboard into the graph and returns the traceID of the new trace.

## SaveTrace

**Command Syntax:** :Displays:Graph(i):SaveTrace *TraceID*, *FileName*, *GraphData*

**Command Argument(s):** *TraceID* <int>

*FileName* <string>

*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Displays:Graph(1):SaveTrace 101, "MyFile.XML", gdNeverS

**Description:** Saves the the trace with the given ID into the specified file. The *GraphData* argument determines whether data will be saved with the trace.

## SelectedTrace?

**Command Syntax:** :Displays:Graph(i):SelectedTrace?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):SelectedTrace] *Trace*

**Response Argument(s):** *Trace* <itemstring>

**Example:** :Displays:Graph(1):SelectedTrace?  
[:Displays:Graph(1):SelectedTrace] "TRACE(1)"

**Description:** Returns a string containing the traceID of the selected trace. Properties of the the selected trace in the example above, for instance, could then be accessed using the object :Displays:Graph(i):Trace(1).

## SelectTrace

**Command Syntax:** :Displays:Graph(i):SelectTrace *Trace*

**Command Argument(s):** *Trace* <itemstring>

**Example:** :Displays:Graph(1):SelectTrace "TRACE(1)"

**Description:** Sets the active trace on the display to be the specified trace.

## TraceCount?

**Command Syntax:** :Displays:Graph(i):TraceCount?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):TraceCount] NumTraces

**Response Argument(s):** NumTraces <int>

**Example:** :Displays:Graph(1):TraceCount?  
[:Displays:Graph(1):TraceCount] 4

**Description:** Returns the total number of traces in the graph.

## TraceItem?

**Command Syntax:** :Displays:Graph(i):TraceItem? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:Displays:Graph(i):TraceItem] Trace

**Response Argument(s):** Trace <itemstring>

**Example:** :Displays:Graph(1):TraceItem? 0  
[:Displays:Graph(1):TraceItem] "TRACE(1)"

**Description:** Returns a string containing the traceID of the Index<sup>th</sup> trace of the graph. Properties of the the selected trace in the example above, for instance, could then be accessed using the object :Displays:Graph(i):Trace(1).

## 2.3.16.1.1 Graph Trace

<b>Object:</b>	<b>:Displays:Graph(i):Trace(j)</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;, j &lt;int&gt;</i>
<i>Description:</i>	Commands related to a single trace of a graph.

## Trace Measurement Commands

### Offline?

**Command Syntax:** :Displays:Graph(i):Trace(j):Offline?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Trace(j):Offline] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Offline?

**[ :Displays:Graph(1):Trace(101):Offline] False**

**Related Command(s):** Offline

**Description:** Returns the Online/Offline status of the trace.

### Offline

**Command Syntax:** :Displays:Graph(i):Trace(j):Offline *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Offline False

**Related Command(s):** Offline?

**Description:** Sets the online/offline status of the trace.

### GetMeasID?

**Command Syntax:** :Displays:Graph(i):Trace(j):GetMeasID?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):Trace(j):GetMeasID] *MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 |

msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 |  
 msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 |  
 msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 |  
 msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 |  
 msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 |  
 msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 |  
 msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 |  
 msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 |  
 msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 |  
 msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |  
 msA0MTthdnBinsB=1205 | msA0MTThdBinsA=1206 | msA0MTThdBinsB=1207 |  
 msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |  
 msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |  
 msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |  
 msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |  
 msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |  
 msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |  
 msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |  
 msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |  
 msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |  
 msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |  
 msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |  
 msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |  
 msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |  
 msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |  
 msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |  
 msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |  
 msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |  
 msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |  
 msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |  
 msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |  
 msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |  
 msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |  
 msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |  
 msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |  
 msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |  
 msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |  
 msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |  
 msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |  
 msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |  
 msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |  
 msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |  
 msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |  
 msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |  
 msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |  
 msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |  
 msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |  
 msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |  
 msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |  
 msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |  
 msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |  
 msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |  
 msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |  
 msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |  
 msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |  
 msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |

```
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}
```

**Example:** `:Displays:Graph(1):Trace(101):GetMeasID?`  
`[ :Displays:Graph(1):Trace(101):GetMeasID] msNull`

**Description:** Returns the measurement ID associated with the trace.

## GetN?

**Command Syntax:** `:Displays:Graph(i):Trace(j):GetN?`

**Command Argument(s):** None

**Response Syntax:** `[ :Displays:Graph(i):Trace(j):GetN] N`

**Response Argument(s):** `N <int>`

**Example:** `:Displays:Graph(1):Trace(101):GetN?`  
`[ :Displays:Graph(1):Trace(101):GetN] 1024`

**Description:** Returns the number of data-points in the trace.

## GetNumStoredData?

**Command Syntax:** `:Displays:Graph(i):Trace(j):GetNumStoredData?`

**Command Argument(s):** None

**Response Syntax:** `[ :Displays:Graph(i):Trace(j):GetNumStoredData] N`

**Response Argument(s):** `N <int>`

**Example:** `:Displays:Graph(1):Trace(101):GetNumStoredData?`  
`[ :Displays:Graph(1):Trace(101):GetNumStoredData] 2`

**Description:** Returns the current storage depth of the trace. This is the number of measurements that have been taken since free-run mode was started. Measurements are stored up to the number set by the "SetStorageDepth" command.

## SetStorageDepth

**Command Syntax:** `:Displays:Graph(i):Trace(j):SetStorageDepth Depth`

**Command Argument(s):** `Depth <int>`

**Example:** `:Displays:Graph(1):Trace(101):SetStorageDepth Value`

**Description:** Sets the storage depth for the trace. For traces displaying scalar data (stripchart type traces) this is the maximum number of points in the trace, for vector (FFT-type) traces this is the number of vector arrays that will be stored for the trace.

## GetStorageDepth?

**Command Syntax:** :Displays:Graph(i):Trace(j):GetStorageDepth?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):Trace(j):GetStorageDepth] *Depth*

**Response Argument(s):** *Depth* <int>

**Example:** :Displays:Graph(1):Trace(101):GetStorageDepth?  
[:Displays:Graph(1):Trace(101):GetStorageDepth] Value

**Description:** Returns the storage depth for the trace. For traces displaying scalar data (stripchart type traces) this is the maximum number of points in the trace, for vector (FFT-type) traces this is the number of vector arrays that will be stored for the trace.

## RecallStoredData

**Command Syntax:** :Displays:Graph(i):Trace(j):RecallStoredData *Index*

**Command Argument(s):** *Index* <int>

**Example:** :Displays:Graph(1):Trace(101):RecallStoredData 0

**Description:** For vector (FFT-type) traces, recalls the data from the Index<sup>th</sup> stored measurement into the trace. An index of 0 corresponds to the oldest stored data.

## RecallStoredDataIntoNewTrace?

**Command Syntax:** :Displays:Graph(i):Trace(j):RecallStoredDataIntoNewTrace? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:Displays:Graph(i):Trace(j):RecallStoredDataIntoNewTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):RecallStoredDataIntoNewTra  
[:Displays:Graph(1):Trace(101):RecallStoredDataIntoNewTr

**Description:** For vector (FFT-type) traces, recalls the data from the Index<sup>th</sup> stored measurement into a new trace and returns the TraceID of the newly created trace..

## GetXArray?

**Command Syntax:** :Displays:Graph(i):Trace(j):GetXArray? *XUnit*

**Command Argument(s):** *XUnit* <string>

**Response Syntax:** [:Displays:Graph(i):Trace(j):GetXArray] *XArray*

**Response Argument(s):** *XArray* <doublearray>

**Example:** :Displays:Graph(1):Trace(101):GetXArray? ""  
[:Displays:Graph(1):Trace(101):GetXArray] "0,1,2,3,4,5,6

**Description:** Returns the X-axis array for the trace in the specified units. The empty string ("") specifies the current X-units of the trace.

## GetXY?

**Command Syntax:** :Displays:Graph(i):Trace(j):GetXY? *Index, XUnit, YUnit*

**Command Argument(s):** *Index* <int>  
*XUnit* <string>  
*YUnit* <string>

**Response Syntax:** [:Displays:Graph(i):Trace(j):GetXY] *XYRdg*

**Response Argument(s):** *XYRdg* <doublearray>

**Example:** :Displays:Graph(1):Trace(101):GetXY? 0, "", ""  
 [:Displays:Graph(1):Trace(101):GetXY] 0,1.21913957733764

**Description:** Returns the X-Y values corresponding to the Index<sup>th</sup> point in the trace in the specified units. The empty string ("") specifies the current X or Y units.

## GetYArray?

**Command Syntax:** :Displays:Graph(i):Trace(j):GetYArray? *YUnit*

**Command Argument(s):** *YUnit* <string>

**Response Syntax:** [:Displays:Graph(i):Trace(j):GetYArray] *YArray*

**Response Argument(s):** *YArray* <doublearray>

**Example:** :Displays:Graph(1):Trace(101):GetYArray? ""  
 [:Displays:Graph(1):Trace(101):GetYArray] 1.219139577337

**Description:** Returns the Y-axis array for the trace in the specified units. The empty string ("") specifies the current Y-units of the trace.

## Scaling and Trace Appearance Commands

### Rename

**Command Syntax:** :Displays:Graph(i):Trace(j):Rename *Name*

**Command Argument(s):** *Name* <string>

**Example:** :Displays:Graph(1):Trace(101):Rename "MyTrace"

**Description:** Changes the name of the trace in the graph trace listing.

### SetColor

**Command Syntax:** :Displays:Graph(i):Trace(j):SetColor *Color*

**Command Argument(s):** *Color* <int>

**Example:** :Displays:Graph(1):Trace(101):SetColor 144

**Description:** Sets the color of the trace.



## SetWidth

**Command Syntax:** :Displays:Graph(i):Trace(j):SetWidth *Width*

**Command Argument(s):** *Width* <int>

**Example:** :Displays:Graph(1):Trace(101):SetWidth 2

**Description:** Sets the width of the trace in pixels.

## xAxis?

**Command Syntax:** :Displays:Graph(i):Trace(j):xAxis?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Trace(j):xAxis] *Value*

**Response Argument(s):** *Value* <int> {xaIndex=0 | xaTime=1 | xaSweep=2}

**Example:** :Displays:Graph(1):Trace(101):xAxis?

**[:Displays:Graph(1):Trace(101):xAxis] xaIndex**

**Related Command(s):** xAxis

**Description:** For stripchart traces, queries the X-axis type.

## xAxis

**Command Syntax:** :Displays:Graph(i):Trace(j):xAxis *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {xaIndex=0 | xaTime=1 | xaSweep=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):xAxis xaIndex

**Related Command(s):** xAxis?

**Description:** For stripchart traces, sets the X-axis type

## Xlog?

**Command Syntax:** :Displays:Graph(i):Trace(j):Xlog?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Trace(j):Xlog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Xlog?

**[:Displays:Graph(1):Trace(101):Xlog] False**

**Related Command(s):** Xlog

**Description:** Queries the log/linear status of the trace X-axis.

## Xlog

**Command Syntax:** :Displays:Graph(i):Trace(j):Xlog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Xlog False

**Related Command(s):** Xlog?

**Description:** Sets the log/linear status of the trace X-axis.

## Xmax?

**Command Syntax:** :Displays:Graph(i):Trace(j):Xmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):Xmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Trace(101):Xmax?  
 [:Displays:Graph(1):Trace(101):Xmax] 40

**Related Command(s):** Xmax

**Description:** Queries the maximum displayed X-axis value.

## Xmax

**Command Syntax:** :Displays:Graph(i):Trace(j):Xmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Xmax 40

**Related Command(s):** Xmax?

**Description:** Sets the maximum displayed X-axis value.

## Xmin?

**Command Syntax:** :Displays:Graph(i):Trace(j):Xmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):Xmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Trace(101):Xmin?  
 [:Displays:Graph(1):Trace(101):Xmin] 0.00000 SEC

**Related Command(s):** Xmin

**Description:** Queries the minimum displayed X-axis value.

## Xmin

**Command Syntax:** :Displays:Graph(i):Trace(j):Xmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Xmin 10.0 SEC

**Related Command(s):** Xmin?

**Description:** Sets the minimum displayed X-axis value.

## xScaleMode?

**Command Syntax:** :Displays:Graph(i):Trace(j):xScaleMode?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Trace(j):xScaleMode] *Value*

**Response Argument(s):** *Value* <int> {xaFixed=0 | xaPan=1 | xaFull=2}

**Example:** :Displays:Graph(1):Trace(101):xScaleMode?  
 [:Displays:Graph(1):Trace(101):xScaleMode] xaFixed

**Related Command(s):** xScaleMode

**Description:** For stripchart type displays, queries the scaling mode used when new points are added to the trace.

## xScaleMode

**Command Syntax:** :Displays:Graph(i):Trace(j):xScaleMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {xaFixed=0 | xaPan=1 | xaFull=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):xScaleMode xaFixed

**Related Command(s):** xScaleMode?

**Description:** For stripchart type displays, sets the scaling mode used when new points are added to the trace.

## Ylog?

**Command Syntax:** :Displays:Graph(i):Trace(j):Ylog?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Trace(j):Ylog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Ylog?  
 [:Displays:Graph(1):Trace(101):Ylog] False

**Related Command(s):** Ylog

**Description:** Queries the log/linear status of the trace Y-axis.

## Ylog

**Command Syntax:** :Displays:Graph(i):Trace(j):Ylog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Ylog False

**Related Command(s):** Ylog?

**Description:** Sets the log/linear status of the trace Y-axis.

## Ymax?

**Command Syntax:** :Displays:Graph(i):Trace(j):Ymax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):Ymax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Trace(101):Ymax?  
 [:Displays:Graph(1):Trace(101):Ymax] 8.95705E-06 VP

**Related Command(s):** Ymax

**Description:** Queries the maximum displayed Y-axis value.

## Ymax

**Command Syntax:** :Displays:Graph(i):Trace(j):Ymax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Ymax 1.0E-5 VP

**Related Command(s):** Ymax?

**Description:** Sets the maximum displayed Y-axis value.

## Ymin?

**Command Syntax:** :Displays:Graph(i):Trace(j):Ymin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):Ymin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Trace(101):Ymin?  
 [:Displays:Graph(1):Trace(101):Ymin] -2.72367E-05 VP

**Related Command(s):** Ymin

**Description:** Queries the minimum displayed Y-axis value.

## Ymin

**Command Syntax:** :Displays:Graph(i):Trace(j):Ymin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Trace(101):Ymin -3.0E-5 VP

**Related Command(s):** Ymin?

**Description:** Sets the minimum displayed Y-axis value.

## AutoScale

**Command Syntax:** :Displays:Graph(i):Trace(j):AutoScale

**Command Argument(s):** None

**Example:** :Displays:Graph(1):Trace(101):AutoScale

**Description:** Autoscales the trace's X and Y axes.

## AutoScaleX

**Command Syntax:** :Displays:Graph(i):Trace(j):AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:Graph(1):Trace(101):AutoScaleX

**Description:** Autoscales the trace X-axis.

## AutoScaleY

**Command Syntax:** :Displays:Graph(i):Trace(j):AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:Graph(1):Trace(101):AutoScaleY

**Description:** Autoscales the trace Y-axis.

## Trace Calculator Commands

### Calc2Sigma?

**Command Syntax:** :Displays:Graph(i):Trace(j):Calc2Sigma? *FromX*, *ToX*

**Command Argument(s):** *FromX* <unit>  
*ToX* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):Calc2Sigma] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):Calc2Sigma? 10 HZ, 100 HZ  
**[:Displays:Graph(1):Trace(101):Calc2Sigma] 104**

**Description:** Creates a trace constant value which exceeds 95% (2-sigma) of the y-values within the specified X-axis range and returns the traceID of the newly created trace.

## CalcAverage?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcAverage? *FromX, ToX, Mode*

**Command Argument(s):** *FromX* <unit>  
*ToX* <unit>  
*Mode* <int> {amLinear=0 | amRMS=1 | amVariance=2}

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcAverage] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcAverage? 10 Hz,100 Hz,  
 [:Displays:Graph(1):Trace(101):CalcAverage] 104

**Description:** Creates a trace containing the average value of the original trace data within the specified X-range (using the specified average method) and returns the traceID of the newly created trace.

## CalcDifference?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcDifference? *Trace2*

**Command Argument(s):** *Trace2* <itemstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcDifference] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcDifference? 102  
 [:Displays:Graph(1):Trace(101):CalcDifference] 104

**Description:** Creates a trace containing the difference between the original trace and the trace referenced by the *Trace2* argument. The query returns the ID of the newly created trace.

## CalcEQ?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcEQ? *EQFilename, Invert*

**Command Argument(s):** *EQFilename* <string>  
*Invert* <bool> {False=0 | True=1}

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcEQ] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcEQ? "MyEq.EQ", False  
 [:Displays:Graph(1):Trace(101):CalcEQ] 104

**Description:** Creates a new trace corresponding to the application of the specified EQ file to the original trace data. The EQ file can be applied normally, or inverted. The return value is the traceID of the newly created trace.

## CalcGroupDelay?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcGroupDelay?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcGroupDelay] CalcTraceID

**Response Argument(s):** CalcTraceID <int>

**Example:** :Displays:Graph(1):Trace(101):CalcGroupDelay?  
[:Displays:Graph(1):Trace(101):CalcGroupDelay] 104

**Description:** Creates a new trace with the group delay corresponding to the original trace data. The original trace must be a phase vs. frequency trace, otherwise the command triggers an execution error. The return value is the traceID of the newly created trace.

## CalcInvert?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcInvert? XPos

**Command Argument(s):** XPos <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcInvert] CalcTraceID

**Response Argument(s):** CalcTraceID <int>

**Example:** :Displays:Graph(1):Trace(101):CalcInvert? 1000 HZ  
[:Displays:Graph(1):Trace(101):CalcInvert] 104

**Description:** Creates an inverted trace around the specified X-position and returns the traceID of the newly created trace.

## CalcLinearity?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcLinearity? FromX, ToX

**Command Argument(s):** FromX <unit>  
ToX <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcLinearity] CalcTraceID

**Response Argument(s):** CalcTraceID <int>

**Example:** :Displays:Graph(1):Trace(101):CalcLinearity? 0 HZ, 10000  
[:Displays:Graph(1):Trace(101):CalcLinearity] 104

**Description:** Calculates the linear fit to the data in the specified X-region and then returns the traceID of a new trace that contains the *difference* between the original data and the linear fit.

## CalcMakeArb?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcMakeArb? *ArbFilename*

**Command Argument(s):** *ArbFilename* <string>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcMakeArb] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcMakeArb? "Myfile.Arb"  
[:Displays:Graph(1):Trace(101):CalcMakeArb] 104

**Description:** Creates an arb file with the data in the current trace. Returns the traceID of a new trace identical to the original trace.

## CalcMakeEQ?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcMakeEQ? *XPos, EQFilename*

**Command Argument(s):** *XPos* <unit>  
*EQFilename* <string>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcMakeEQ] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcMakeEQ? 1000 HZ, "MyEq."  
[:Displays:Graph(1):Trace(101):CalcMakeEQ] 104

**Description:** Normalizes the trace data to the value at the specified X-axis position and creates an EQ file corresponding to the normalized data. The query returns the traceID of a new trace containing the normalized data. Causes an execution error if the original trace does not have a frequency X-axis.

## CalcMaximum?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcMaximum? *FromX, ToX*

**Command Argument(s):** *FromX* <unit>  
*ToX* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcMaximum] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcMaximum? 0 HZ, 10000 H  
[:Displays:Graph(1):Trace(101):CalcMaximum] 104

**Description:** Returns the traceID of a newly created trace containing the maximum value of the original trace.



## CalcMinimum?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcMinimum? *FromX, ToX*

**Command Argument(s):** *FromX* <unit>  
*ToX* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcMinimum] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcMinimum? 0 HZ, 10000 H  
[:Displays:Graph(1):Trace(101):CalcMinimum] 104

**Description:** Returns the traceID of a newly created trace containing the minimum value of the original trace.

## CalcMultiply?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcMultiply? *Trace2*

**Command Argument(s):** *Trace2* <itemstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcMultiply] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcMultiply? 105  
[:Displays:Graph(1):Trace(101):CalcMultiply] 106

**Description:** Returns the traceID of a new trace containing the product of the original trace with the trace specified by the *Trace2* argument.

## CalcNormalize?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcNormalize? *XPos, YVal*

**Command Argument(s):** *XPos* <unit>  
*YVal* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcNormalize] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcNormalize? 200 HZ, 1 V  
[:Displays:Graph(1):Trace(101):CalcNormalize] 104

**Description:** Returns the traceID of a new trace containing the data in the original trace normalized to the value of *YVal* at the X position *Xpos*.

## CalcParametric?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcParametric? *Trace2*

**Command Argument(s):** *Trace2* <itemstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcParametric] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcParametric? Value  
[:Displays:Graph(1):Trace(101):CalcParametric] Value

**Description:** Returns the traceID of a new trace containing the Y data in the original trace as the Y data, and the Y data in the *Trace2* as the X data. *Trace2* must be at least as long of the original trace.

## CalcRatio?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcRatio? *Trace2*

**Command Argument(s):** *Trace2* <itemstring>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcRatio] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcRatio? 103  
[:Displays:Graph(1):Trace(101):CalcRatio] 104

**Description:** Returns the traceID of a new trace whose data is the ratio of the original trace to the trace specified in the *Trace2* argument.

## CalcSmooth?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcSmooth? *Mode, SmoothParam, Repeat*

**Command Argument(s):** *Mode* <int> {saBoxAvg=0 | saBinomial=1 | saSavitzkyGolay=2}  
*SmoothParam* <int>  
*Repeat* <int>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcSmooth] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcSmooth? saBoxAvg, Valu  
[:Displays:Graph(1):Trace(101):CalcSmooth] Value

**Description:** Returns the traceID of a new trace whose data is a smoothed version of the original trace calculated with the specified smoothing algorithm.

## CalcTrim?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcTrim? *FromX, ToX*

**Command Argument(s):** *FromX* <unit>  
*ToX* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcTrim] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcTrim? 100 HZ, 1000 HZ  
[:Displays:Graph(1):Trace(101):CalcTrim] 104

**Description:** Returns the traceID of a new trace containing the original trace data but only including points within the specified X-range.

## CalcUnwrap?

**Command Syntax:** :Displays:Graph(i):Trace(j):CalcUnwrap? *YTolerance*

**Command Argument(s):** *YTolerance* <unit>

**Response Syntax:** [:Displays:Graph(i):Trace(j):CalcUnwrap] *CalcTraceID*

**Response Argument(s):** *CalcTraceID* <int>

**Example:** :Displays:Graph(1):Trace(101):CalcUnwrap? 180

**[ :Displays:Graph(1):Trace(101):CalcUnwrap] 104**

**Description:** Returns the traceID of a new trace containing the "unwrapped" version of the original phase curve. Causes an execution error if the original trace is not phase vs. frequency.

## 2.3.16.1.2 Graph Cursor

<b>Object:</b>	<b>:Displays:Graph(i):Cursor</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the graph cursor.

**Enabled?**

**Command Syntax:** :Displays:Graph(i):Cursor:Enabled?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:Enabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Enabled?  
[:Displays:Graph(1):Cursor:Enabled] False

**Related Command(s):** Enabled

**Description:** Queries whether graph cursors are on.

**Enabled**

**Command Syntax:** :Displays:Graph(i):Cursor:Enabled *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Enabled False

**Related Command(s):** Enabled?

**Description:** Sets whether graph cursors are on.

**x1?**

**Command Syntax:** :Displays:Graph(i):Cursor:x1? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:x1] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:x1?  
[:Displays:Graph(1):Cursor:x1] 199.219 HZ

**Related Command(s):** x1

**Description:** Queries the cursor 1 X-axis position.

## x1

**Command Syntax:** :Displays:Graph(i):Cursor:x1 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Graph(1):Cursor:x1 300.33 HZ

**Related Command(s):** x1?

**Description:** Sets the cursor 1 X-axis position.

## x2?

**Command Syntax:** :Displays:Graph(i):Cursor:x2? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:x2] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:x2?  
[:Displays:Graph(1):Cursor:x2] 2.78906 HZ

**Related Command(s):** x2

**Description:** Queries the cursor 2 X-axis position.

## x2

**Command Syntax:** :Displays:Graph(i):Cursor:x2 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Graph(1):Cursor:x2 *Value*

**Related Command(s):** x2?

**Description:** Sets the cursor 2 X-axis position.

## y1Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:y1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:y1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:y1Rdg?  
[:Displays:Graph(1):Cursor:y1Rdg] 57.3 DBVRMS

**Description:** Queries the cursor 1 Y-value..

## y2Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:y2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:y2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:y2Rdg?

**[ :Displays:Graph(1):Cursor:y2Rdg ] 14.8942 DBVRMS**

**Description:** Queries the cursor 2 Y-value.

## dxCalc?

**Command Syntax:** :Displays:Graph(i):Cursor:dxCalc?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:dxCalc] *Value*

**Response Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatioDB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

**Example:** :Displays:Graph(1):Cursor:dxCalc?

**[ :Displays:Graph(1):Cursor:dxCalc ] cdDelta**

**Related Command(s):** dxCalc

**Description:** Queries the method for calculating the X-axis difference.

## dxCalc

**Command Syntax:** :Displays:Graph(i):Cursor:dxCalc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatioDB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:dxCalc cdDelta

**Related Command(s):** dxCalc?

**Description:** Sets the method for calculating the X-axis difference.

## dxRdg?

**Command Syntax:** :Displays:Graph(i):Cursor:dxRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:dxRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:dxRdg?

**[ :Displays:Graph(1):Cursor:dxRdg ] 2589.84 HZ**

**Description:** Queries the cursor1/cursor2 X-axis difference.

## GetDxUnit?

**Command Syntax:** :Displays:Graph(i):Cursor:GetDxUnit?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):Cursor:GetDxUnit] *dxUnit*

**Response Argument(s):** *dxUnit* <string>

**Example:** :Displays:Graph(1):Cursor:GetDxUnit?  
[:Displays:Graph(1):Cursor:GetDxUnit] HZ

**Description:** Queries the units of the delta-X value.

## dyCalc?

**Command Syntax:** :Displays:Graph(i):Cursor:dyCalc?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:dyCalc] *Value*

**Response Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatioDB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

**Example:** :Displays:Graph(1):Cursor:dyCalc?  
[:Displays:Graph(1):Cursor:dyCalc] cdDelta

**Related Command(s):** dyCalc

**Description:** Queries the method used to calculate the Y-axis difference.

## dyCalc

**Command Syntax:** :Displays:Graph(i):Cursor:dyCalc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatioDB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:dyCalc cdDelta

**Related Command(s):** dyCalc?

**Description:** Sets the method used to calculate the Y-axis difference.

## dyRdg?

**Command Syntax:** :Displays:Graph(i):Cursor:dyRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:dyRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:dyRdg?  
[:Displays:Graph(1):Cursor:dyRdg] -42.4058 DB

**Description:** Queries the delta Y value.

## GetDyUnit?

**Command Syntax:** :Displays:Graph(i):Cursor:GetDyUnit?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Graph(i):Cursor:GetDyUnit] *dyUnit*

**Response Argument(s):** *dyUnit* <string>

**Example:** :Displays:Graph(1):Cursor:GetDyUnit?

[ :Displays:Graph(1):Cursor:GetDyUnit ] DBVRMS

**Description:** Queries the units of the delta-Y reading.



## MoveToMax

**Command Syntax:** :Displays:Graph(i):Cursor:MoveToMax *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:MoveToMax ciCursor1

**Description:** Moves the specified cursor to the position corresponding to the maximum Y-value in the trace.

## MoveToMin

**Command Syntax:** :Displays:Graph(i):Cursor:MoveToMin *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:MoveToMin ciCursor1

**Description:** Moves the specified cursor to the position corresponding to the minimum Y-value within in the trace.

## MoveToPeakL

**Command Syntax:** :Displays:Graph(i):Cursor:MoveToPeakL *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:MoveToPeakL ciCursor1

**Description:** Moves the specified cursor to the next peak left.

## MoveToPeakR

**Command Syntax:** :Displays:Graph(i):Cursor:MoveToPeakR *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:MoveToPeakR ciCursor1

**Description:** Moves the specified cursor to the next peak right.





## Avg1Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Avg1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Avg1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Avg1Rdg?  
[:Displays:Graph(1):Cursor:Avg1Rdg] 57.311

**Description:** Queries the Average value of cursor 1.

## Avg2Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Avg2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Avg2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Avg2Rdg?  
[:Displays:Graph(1):Cursor:Avg2Rdg] 57.311

**Description:** Queries the Average value of cursor 2.

## Max1Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Max1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Max1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Max1Rdg?  
[:Displays:Graph(1):Cursor:Max1Rdg] 57.397

**Description:** Queries the Maximum value of cursor 1.

## Max2Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Max2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Max2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Max2Rdg?  
[:Displays:Graph(1):Cursor:Max2Rdg] 57.397

**Description:** Queries the Maximum value of cursor 2.

## Min1Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Min1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Min1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Min1Rdg?  
[:Displays:Graph(1):Cursor:Min1Rdg] 57.247

**Description:** Queries the minimum value of cursor 1.

## Min2Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Min2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Min2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Min2Rdg?  
[:Displays:Graph(1):Cursor:Min2Rdg] 57.247

**Description:** Queries the minimum value of cursor 2.

## Sigma1Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Sigma1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Sigma1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Sigma1Rdg?  
[:Displays:Graph(1):Cursor:Sigma1Rdg] 0.037555

**Description:** Queries the standard deviation of cursor 1.

## Sigma2Rdg?

**Command Syntax:** :Displays:Graph(i):Cursor:Sigma2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:Sigma2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:Sigma2Rdg?  
[:Displays:Graph(1):Cursor:Sigma2Rdg] 0.037555

**Description:** Queries the standard deviation of cursor 2.

## ResetStats

**Command Syntax:** :Displays:Graph(i):Cursor:ResetStats *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:ResetStats ciCursor1

**Description:** Resets the cursor statistics calculation for the specified cursor.

## StartStats

**Command Syntax:** :Displays:Graph(i):Cursor:StartStats *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:StartStats ciCursor1

**Description:** Starts the cursor statistics calculation for the specified cursor.

## StopStats

**Command Syntax:** :Displays:Graph(i):Cursor:StopStats *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:StopStats ciCursor1

**Description:** Stops cursor statistics calculations for the specified cursor.

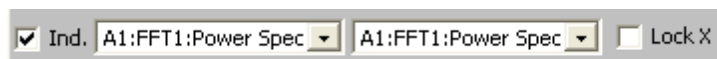
## ToggleStats

**Command Syntax:** :Displays:Graph(i):Cursor:ToggleStats *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:Graph(1):Cursor:ToggleStats ciCursor1

**Description:** Toggles the start/stopped state of cursor statistics calculation for the specified cursor.



## Cursor1Trace?

**Command Syntax:** :Displays:Graph(i):Cursor:Cursor1Trace?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:Cursor1Trace] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Displays:Graph(1):Cursor:Cursor1Trace?

**[ :Displays:Graph(1):Cursor:Cursor1Trace] Value**

**Related Command(s):** Cursor1Trace

**Description:** Returns an index into the trace list corresponding to the trace associated with cursor 1.

## Cursor1Trace

**Command Syntax:** :Displays:Graph(i):Cursor:Cursor1Trace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Cursor1Trace *Value*

**Related Command(s):** Cursor1Trace?

**Description:** Sets the trace associated with cursor 1. The *Value* argument is a 0-based index into the trace list for the graph. The trace associated with the cursors can only be changed when "Independent Cursors" is on. Otherwise the cursors are always associated with the active trace.

## Cursor2Trace?

**Command Syntax:** :Displays:Graph(i):Cursor:Cursor2Trace?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:Cursor2Trace] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Displays:Graph(1):Cursor:Cursor2Trace?  
 [:Displays:Graph(1):Cursor:Cursor2Trace] 2

**Related Command(s):** Cursor2Trace

**Description:** Returns an index into the trace list corresponding to the trace associated with cursor 2.

## Cursor2Trace

**Command Syntax:** :Displays:Graph(i):Cursor:Cursor2Trace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Cursor2Trace *Value*

**Related Command(s):** Cursor2Trace?

**Description:** Sets the trace associated with cursor 2. The *Value* argument is a 0-based index into the trace list for the graph. The trace associated with the cursors can only be changed when "Independent Cursors" is on. Otherwise the cursors are always associated with the active trace.

## Independent?

**Command Syntax:** :Displays:Graph(i):Cursor:Independent?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:Independent] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Independent?

[ :Displays:Graph(1):Cursor:Independent] False

**Related Command(s):** Independent

**Description:** Queries the independent cursor status.

## Independent

**Command Syntax:** :Displays:Graph(i):Cursor:Independent *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Independent False

**Related Command(s):** Independent?

**Description:** Sets the independent cursor status.

## Xlock?

**Command Syntax:** :Displays:Graph(i):Cursor:Xlock?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:Xlock] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Xlock?

[ :Displays:Graph(1):Cursor:Xlock] False

**Related Command(s):** Xlock

**Description:** When independent cursors are on, queries whether the two cursors X-values are locked together.

## Xlock

**Command Syntax:** :Displays:Graph(i):Cursor:Xlock *Value* [, *AllowCoercion*]

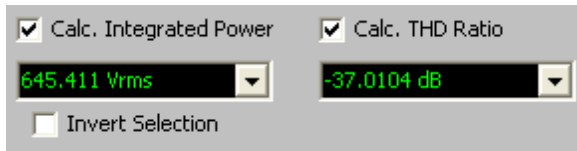
**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:Xlock False

**Related Command(s):** Xlock?

**Description:** When independent cursors are on, sets whether the two cursors X-values are locked together.



## CalcIntegratedPwr?

**Command Syntax:** :Displays:Graph(i):Cursor:CalcIntegratedPwr?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:CalcIntegratedPwr] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:CalcIntegratedPwr?

[ :Displays:Graph(1):Cursor:CalcIntegratedPwr] False

**Related Command(s):** CalcIntegratedPwr

**Description:** Queries the on/off status of integrated power calculation.

## CalcIntegratedPwr

**Command Syntax:** :Displays:Graph(i):Cursor:CalcIntegratedPwr Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:CalcIntegratedPwr False

**Related Command(s):** CalcIntegratedPwr?

**Description:** Sets the on/off status of integrated power calculation.

## CalcTHD?

**Command Syntax:** :Displays:Graph(i):Cursor:CalcTHD?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:CalcTHD] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:CalcTHD?

[ :Displays:Graph(1):Cursor:CalcTHD] False

**Related Command(s):** CalcTHD

**Description:** Queries the on/off status of cursor THD calculation.

## CalcTHD

**Command Syntax:** :Displays:Graph(i):Cursor:CalcTHD *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:CalcTHD False

**Related Command(s):** CalcTHD?

**Description:** Sets the on/off status of cursor THD calculation.

## IntegratedPwrInvertSel?

**Command Syntax:** :Displays:Graph(i):Cursor:IntegratedPwrInvertSel?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Cursor:IntegratedPwrInvertSel] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:IntegratedPwrInvertSel?  
[:Displays:Graph(1):Cursor:IntegratedPwrInvertSel] False

**Related Command(s):** IntegratedPwrInvertSel

**Description:** Queries the inverted/non-inverted status of cursor integrated power selection. When non-inverted the integrated power is calculated between the two cursors. When inverted the power is calculated for all values *except* those between the two cursors.

## IntegratedPwrInvertSel

**Command Syntax:** :Displays:Graph(i):Cursor:IntegratedPwrInvertSel *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Cursor:IntegratedPwrInvertSel False

**Related Command(s):** IntegratedPwrInvertSel?

**Description:** Sets the inverted/non-inverted status of cursor integrated power selection. When non-inverted the integrated power is calculated between the two cursors. When inverted the power is calculated for all values *except* those between the two cursors.

## IntegratedPwrRdg?

**Command Syntax:** :Displays:Graph(i):Cursor:IntegratedPwrRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:IntegratedPwrRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:IntegratedPwrRdg? "VRMS"  
[:Displays:Graph(1):Cursor:IntegratedPwrRdg] 2.14228E-06

**Description:** Returns the result of the integrated power calculation in the specified units.

## THDRdg?

**Command Syntax:** :Displays:Graph(i):Cursor:THDRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Graph(i):Cursor:THDRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Graph(1):Cursor:THDRdg?

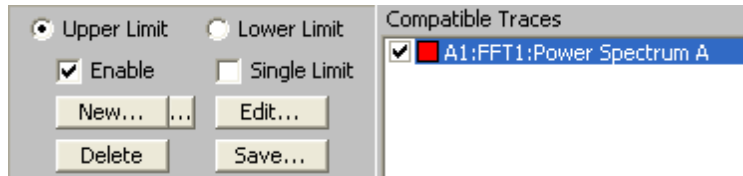
**[ :Displays:Graph(1):Cursor:THDRdg] -37.0104 DB**

**Description:** Returns the cursor THD ratio calculation. The low-frequency cursor is used as the fundamental, and all harmonics are included up to and including the value of the high-frequency cursor.



2.3.16.1.3 Graph Limit

<b>Object:</b>	<b>:Displays:Graph(i):Limit</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;</i>
<i>Description:</i>	Commands related to graph limit-testing.



### LimitFailedRdg?

**Command Syntax:** :Displays:Graph(i):Limit:LimitFailedRdg?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Limit:LimitFailedRdg] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LimitFailedRdg?

**[ :Displays:Graph(1):Limit:LimitFailedRdg ] False**

**Description:** Returns True if any of the traces currently enabled for limit testing fail their limit tests. Otherwise returns false.

### LoLimitEnabled?

**Command Syntax:** :Displays:Graph(1):Limit:LoLimitEnabled?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Limit:LoLimitEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LoLimitEnabled?

**[ :Displays:Graph(1):Limit:LoLimitEnabled ] False**

**Related Command(s):** LoLimitEnabled

**Description:** Queries the enabled/disabled status of the lower limit for the graph.

### LoLimitEnabled

**Command Syntax:** :Displays:Graph(i):Limit:LoLimitEnabled *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LoLimitEnabled False

**Related Command(s):** LoLimitEnabled?

**Description:** Sets the enabled/disabled status of the lower limit for the graph.

## LoLimitExists?

**Command Syntax:** :Displays:Graph(i):Limit:LoLimitExists?

**Response Syntax:** [:Displays:Graph(i):Limit:LoLimitExists] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LoLimitExists?

**[:Displays:Graph(1):Limit:LoLimitExists] False**

**Description:** Returns True if the graph has a configured lower limit.

## LoLimitSingle?

**Command Syntax:** :Displays:Graph(i):Limit:LoLimitSingle?

**Command Argument(s):**

**Response Syntax:** [:Displays:Graph(i):Limit:LoLimitSingle] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LoLimitSingle?

**[:Displays:Graph(1):Limit:LoLimitSingle] False**

**Related Command(s):** LoLimitSingle

**Description:** Returns True if the graph lower limit is a single value that applies to all X-values. Returns false if the lower limit is a multi-segment limit.

## LoLimitSingle

**Command Syntax:** :Displays:Graph(i):Limit:LoLimitSingle *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:LoLimitSingle False

**Related Command(s):** LoLimitSingle?

**Description:** Sets if the graph lower limit type to a single value (Value=True) or a multi-segment limit (Value=False).

## UpLimitEnabled?

**Command Syntax:** :Displays:Graph(i):Limit:UpLimitEnabled?

**Response Syntax:** [:Displays:Graph(i):Limit:UpLimitEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Graph(1):Limit:UpLimitEnabled?

**[:Displays:Graph(1):Limit:UpLimitEnabled] False**

**Related Command(s):** UpLimitEnabled

**Description:** Queries the enabled/disabled status of the upper limit for the graph.

## UpLimitEnabled

**Command Syntax:** `:Displays:Graph(i):Limit:UpLimitEnabled Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Limit:UpLimitEnabled False`

**Related Command(s):** UpLimitEnabled?

**Description:** Sets the enabled/disabled status of the upper limit for the graph..

## UpLimitExists?

**Command Syntax:** `:Displays:Graph(i):Limit:UpLimitExists?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:Graph(i):Limit:UpLimitExists] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Limit:UpLimitExists?`  
`[:Displays:Graph(1):Limit:UpLimitExists] False`

**Description:** Returns True if an upper graph limit currently exists.

## UpLimitSingle?

**Command Syntax:** `:Displays:Graph(i):Limit:UpLimitSingle?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:Graph(i):Limit:UpLimitSingle] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Limit:UpLimitSingle?`  
`[:Displays:Graph(1):Limit:UpLimitSingle] False`

**Related Command(s):** UpLimitSingle

**Description:** Returns true if the upper limit is a single valued limit. Returns false if the upper limit is a multi-segment limit.

## UpLimitSingle

**Command Syntax:** `:Displays:Graph(i):Limit:UpLimitSingle Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:Graph(1):Limit:UpLimitSingle False`

**Related Command(s):** UpLimitSingle?

**Description:** Sets the upper limit to a single value (True) or multi-segment (False).

## DelLimit

**Command Syntax:** :Displays:Graph(i):Limit:DelLimit *LimitID*

**Command Argument(s):** *LimitID* <int> {lidUpLimit=0 | lidLoLimit=1}

**Example:** :Displays:Graph(1):Limit:DelLimit lidUpLimit

**Description:** Deletes the specified graph limit.

## NewLimit?

**Command Syntax:** :Displays:Graph(i):Limit:NewLimit? *LimitID*, *TraceID*

**Command Argument(s):** *LimitID* <int> {lidUpLimit=0 | lidLoLimit=1}  
*TraceID* <int>

**Response Syntax:** [:Displays:Graph(i):Limit:NewLimit] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):Limit:NewLimit? lidUpLimit, 106  
[:Displays:Graph(1):Limit:NewLimit] 107

**Description:** Returns the ID of a newly created limit of the specified type (upper or lower) based on the trace specified in the TraceID argument.

## EditLimit

**Command Syntax:** :Displays:Graph(i):Limit:EditLimit *LimitID*, *XData*, *XUnit*, *YData*, *YUnit*

**Command Argument(s):** *LimitID* <int> {lidUpLimit=0 | lidLoLimit=1}  
*XData* <doublearray>  
*XUnit* <string>  
*YData* <doublearray>  
*YUnit* <string>

**Example:** :Displays:Graph(1):Limit:EditLimit 0, "20,1000,10000", "H"

**Description:** Replaces the existing upper or lower limit with the limits specified in the XData and YData arrays.

## SaveLimit

**Command Syntax:** :Displays:Graph(i):Limit:SaveLimit *LimitID*, *FileName*

**Command Argument(s):** *LimitID* <int> {lidUpLimit=0 | lidLoLimit=1}  
*FileName* <string>

**Example:** :Displays:Graph(1):Limit:SaveLimit lidUpLimit, "MyLimit."

**Description:** Saves the upper or lower limit to the specified filename.

## LoadLimit?

**Command Syntax:** :Displays:Graph(i):Limit:LoadLimit? *LimitID*, *FileName*

**Command Argument(s):** *LimitID* <int> {lidUpLimit=0 | lidLoLimit=1}  
*FileName* <string>

**Response Syntax:** [:Displays:Graph(i):Limit:LoadLimit] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Graph(1):Limit:LoadLimit? lidUpLimit, "MyLimit  
[:Displays:Graph(1):Limit:LoadLimit] 106

**Description:** Loads the limit contained in the specified file and returns the traceID of the limit trace.

## 2.3.16.2 Digitizer Display

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the Digitizer Display.

**ActiveChart?**

**Command Syntax:** :Displays:DigitizerDisplay(i):ActiveChart?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):ActiveChart] *Value*

**Response Argument(s):** *Value* <int> {ddTimeRec=0 | ddSpectrum=1 | ddProbability=2 | ddEyeDiagram=3}

**Example:** :Displays:DigitizerDisplay(1):ActiveChart?

**[ :Displays:DigitizerDisplay(1):ActiveChart ] ddTimeRec**

**Related Command(s):** ActiveChart

**Description:** Queries the active tab of the digitizer display.

**ActiveChart**

**Command Syntax:** :Displays:DigitizerDisplay(i):ActiveChart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ddTimeRec=0 | ddSpectrum=1 | ddProbability=2 | ddEyeDiagram=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ActiveChart ddTimeRec

**Related Command(s):** ActiveChart?

**Description:** Sets the active tab of the digitizer display.

ERS Digitizer Display 0

## Title?

**Command Syntax:** `:Displays:DigitizerDisplay(i):Title?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:DigitizerDisplay(i):Title] Value`

**Response Argument(s):** `Value <string>`

**Example:** `:Displays:DigitizerDisplay(1):Title?`

`[:Displays:DigitizerDisplay(1):Title] Digitizer Display`

**Related Command(s):** Title

**Description:** Returns the title of the digitizer display.

## Title

**Command Syntax:** `:Displays:DigitizerDisplay(i):Title Value [, AllowCoercion]`

**Command Argument(s):** `Value <string>`

`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:DigitizerDisplay(1):Title "My New Title"`

**Related Command(s):** Title?

**Description:** Sets the title of the digitizer display.

## BackgroundColor?

**Command Syntax:** `:Displays:DigitizerDisplay(i):BackgroundColor?`

**Command Argument(s):**

**Response Syntax:** `[:Displays:DigitizerDisplay(i):BackgroundColor] Value`

**Response Argument(s):** `Value <int> {bcWhite=0 | bcBlack=1}`

**Example:** `:Displays:DigitizerDisplay(1):BackgroundColor?`

`[:Displays:DigitizerDisplay(1):BackgroundColor] bcWhite`

**Related Command(s):** BackgroundColor

**Description:** Queries the background color of the display.

## BackgroundColor

**Command Syntax:** `:Displays:DigitizerDisplay(i):BackgroundColor Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {bcWhite=0 | bcBlack=1}`

`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:DigitizerDisplay(1):BackgroundColor bcWhite`

**Related Command(s):** BackgroundColor?

**Description:** Sets the background color of the display.



## Save

**Command Syntax:** :Displays:DigitizerDisplay(i):Save *FileName*, *GraphData*

**Command Argument(s):** *FileName* <string>

*GraphData* <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Displays:DigitizerDisplay(1):Save "MyDigDisp.xml", gdAl

**Description:** Saves the digitizer display to the named file with the specified data options.



## AutoscaleOnAcquire?

**Command Syntax:** :Displays:DigitizerDisplay(i):AutoscaleOnAcquire?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):AutoscaleOnAcquire] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):AutoscaleOnAcquire?

**[ :Displays:DigitizerDisplay(1):AutoscaleOnAcquire] False**

**Related Command(s):** AutoscaleOnAcquire

**Description:** Queries whether the displays will autoscale when a new digitizer record is acquired.

## AutoscaleOnAcquire

**Command Syntax:** :Displays:DigitizerDisplay(i):AutoscaleOnAcquire *Value* [, *AllowCoercion*]

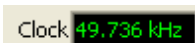
**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):AutoscaleOnAcquire False

**Related Command(s):** AutoscaleOnAcquire?

**Description:** Sets whether the displays will autoscale when a new digitizer record is acquired.



## ClockRdg?

**Command Syntax:** :Displays:DigitizerDisplay(i):ClockRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ClockRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ClockRdg?

**[ :Displays:DigitizerDisplay(1):ClockRdg] 49736 HZ**

**Description:** Returns the digital audio sampling clock frequency recovered from the digitizer record.





## CursorEnabled?

**Command Syntax:** :Displays:DigitizerDisplay(i):CursorEnabled?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):CursorEnabled] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):CursorEnabled?  
[:Displays:DigitizerDisplay(1):CursorEnabled] **False**

**Related Command(s):** CursorEnabled

**Description:** Returns the on/off state of the digitizer display cursor.

## CursorEnabled

**Command Syntax:** :Displays:DigitizerDisplay(i):CursorEnabled *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):CursorEnabled **False**

**Related Command(s):** CursorEnabled?

**Description:** Sets the on/off state of the digitizer display cursor.



## Maximize?

**Command Syntax:** :Displays:DigitizerDisplay(i):Maximize?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):Maximize] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):Maximize?  
[:Displays:DigitizerDisplay(1):Maximize] **False**

**Related Command(s):** Maximize

**Description:** Returns the maximized status of the display. When the display is maximized the main graph area expands to fill the are occupied by the scaling controls and trace listing.

## Maximize

**Command Syntax:** :Displays:DigitizerDisplay(i):Maximize *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):Maximize **False**

**Related Command(s):** Maximize?

**Description:** Sets the maximized status of the display. When the display is maximized the main graph area expands to fill the are occupied by the scaling controls and trace listing.



## Online?

**Command Syntax:** :Displays:DigitizerDisplay(i):Online?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):Online] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):Online?

**[ :Displays:DigitizerDisplay(1):Online ] False**

**Related Command(s):** Online

**Description:** Queries the online/offline status of the display.

## Online

**Command Syntax:** :Displays:DigitizerDisplay(i):Online Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):Online False

**Related Command(s):** Online?

**Description:** Sets the online/offline status of the display.



## Close

**Command Syntax:** :Displays:DigitizerDisplay(i):Close

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):Close

**Description:** Closes the digitizer display form.



## ExportData

**Command Syntax:** :Displays:DigitizerDisplay(i):ExportData FileName

**Command Argument(s):** FileName <string>

**Example:** :Displays:DigitizerDisplay(1):ExportData "MyData.txt"

**Description:** Exports the data on the active page of the digitizer display to a text file with the specified filename.

## ExportGraph

**Command Syntax:** :Displays:DigitizerDisplay(i):ExportGraph FileType, FileName

**Command Argument(s):** FileType <int> {0=BitMap, 1=Enhanced MetaFile, 2=JPEG}

FileName <string>

**Example:** :Displays:DigitizerDisplay(1):ExportGraph 2, "MyGraph.JPG"

**Description:** Exports the graph to the specified type of graphics file with the given filename.



## Print

**Command Syntax:** :Displays:DigitizerDisplay(i):Print

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):Print

**Description:** Prints the digitizer display to the currently configured windows printer.

## 2.3.16.2.1 Digitizer Display Time Record

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): TimeRecChart</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Time Record tab of the Digitizer Display

	min	max
X	-99.313 psec	102.34 $\mu$ sec
Y	-3.3825 V	3.3825 V

## InputYmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:InputYmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:InputYmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:InputYmax?  
[:Displays:DigitizerDisplay(1):TimeRecChart:InputYmax] 1

**Related Command(s):** InputYmax

**Description:** Queries the maximum Y-axis value for the input trace.

## InputYmax

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:InputYmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:InputYmax 1.0

**Related Command(s):** InputYmax?

**Description:** Sets the maximum Y-axis value for the input trace..

## InputYmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:InputYmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:InputYmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:InputYmin?  
[:Displays:DigitizerDisplay(1):TimeRecChart:InputYmin] -

**Related Command(s):** InputYmin

**Description:** Queries the minimum Y-axis value for the input trace.

## InputYmin

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:InputYmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:InputYmin -1.

**Related Command(s):** InputYmin?

**Description:** Sets the minimum Y-axis value for the input trace.

## JitterYmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:JitterYmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:JitterYmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:JitterYmax?  
[:Displays:DigitizerDisplay(1):TimeRecChart:JitterYmax]

**Related Command(s):** JitterYmax

**Description:** Queries the maximum Y-axis value for the jitter trace.

## JitterYmax

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:JitterYmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:JitterYmax 5.

**Related Command(s):** JitterYmax?

**Description:** Sets the maximum Y-axis value for the input trace.

## JitterYmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:JitterYmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:JitterYmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:JitterYmin?  
[:Displays:DigitizerDisplay(1):TimeRecChart:JitterYmin]

**Related Command(s):** JitterYmin

**Description:** Queries the minimum Y-axis value for the jitter trace.

## JitterYmin

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:JitterYmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:JitterYmin -5

**Related Command(s):** JitterYmin?

**Description:** Sets the minimum Y-axis value for the jitter trace..

## Xmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Xmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Xmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Xmax?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Xmax] 0.0001

**Related Command(s):** Xmax

**Description:** Queries the maximum X-axis value for the time record graph.

## Xmax

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Xmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Xmax 103.34E-

**Related Command(s):** Xmax?

**Description:** Sets the maximum X-axis value for the time record graph.

## Xmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Xmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Xmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Xmin?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Xmin] 0.0 S

**Related Command(s):** Xmin

**Description:** Queries the minimum X-axis value for the time record graph.

## Xmin

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Xmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Xmin 0

**Related Command(s):** Xmin?

**Description:** Sets the minimum X-axis value for the time record graph.

## AutoScale

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:AutoScale

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:AutoScale

**Description:** Autoscales the X and Y axes of the time record graph.

## AutoScaleX

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:AutoScaleX

**Description:** Autoscales the X axis of the time record graph.

## AutoScaleY

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:AutoScaleY

**Description:** Autoscales the Y axis of the time record graph.

## ShiftX

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:ShiftX *Direction*

**Command Argument(s):** *Direction* <bool> {xLeft=0|xRight=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:ShiftX xLeft

**Description:** Shifts the X-axis of the time record graph left or right.

## ShiftY

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:ShiftY *Direction*

**Command Argument(s):** *Direction* <bool> {yDown=0|yUp=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:ShiftY yDown

**Description:** Shifts the Y-axis of the time record graph up or down.

## ZoomX

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:ZoomX *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmIn=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:ZoomX zmOut

**Description:** Zoom the X-axis in or out.

## ZoomY

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:ZoomY *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmIn=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:ZoomY zmOut

**Description:** Zoom the Y-axis in or out.



## SelectedTrace?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:SelectedTrace?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:SelectedTrace] *Value*

**Response Argument(s):** *Value* <int> {trInput=0|trJitter=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:SelectedTrace  
[:Displays:DigitizerDisplay(1):TimeRecChart:SelectedTrac

**Related Command(s):** SelectedTrace

**Description:** Queries the selected trace or the time record graph.



## SelectedTrace

**Command Syntax:** `:Displays:DigitizerDisplay(i):TimeRecChart:SelectedTrace Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {trInput=0 | trJitter=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:Displays:DigitizerDisplay(1):TimeRecChart:SelectedTrace`

**Related Command(s):** SelectedTrace?

**Description:** Sets the selected trace for the time record graph..

## IsTraceChecked?

**Command Syntax:** `:Displays:DigitizerDisplay(i):TimeRecChart:IsTraceChecked? Trace`

**Command Argument(s):** `Trace <int> {trInput=0 | trJitter=1}`

**Response Syntax:** `[:Displays:DigitizerDisplay(i):TimeRecChart:IsTraceChecked] Checked`

**Response Argument(s):** `Checked <bool> {False=0 | True=1}`

**Example:** `:Displays:DigitizerDisplay(1):TimeRecChart:IsTraceChecked`  
`[:Displays:DigitizerDisplay(1):TimeRecChart:IsTraceChecked]`

**Description:** Queries whether the speceified trace (Input or Jitter) is checked.

## TraceCheck

**Command Syntax:** `:Displays:DigitizerDisplay(i):TimeRecChart:TraceCheck Trace, Check`

**Command Argument(s):** `Trace <int> {trInput=0 | trJitter=1}`  
`Check <bool> {False=0 | True=1}`

**Example:** `:Displays:DigitizerDisplay(1):TimeRecChart:TraceCheck tr`

**Description:** Sets whether the speceified trace (Input or Jitter) is checked.

## SetColor

**Command Syntax:** `:Displays:DigitizerDisplay(i):TimeRecChart:SetColor Trace, Color`

**Command Argument(s):** `Trace <int> {trInput=0 | trJitter=1}`  
`Color <int>`

**Example:** `:Displays:DigitizerDisplay(1):TimeRecChart:SetColor trIn`

**Description:** Sets the color of the specified trace.

## SetWidth

**Command Syntax:** `:Displays:DigitizerDisplay(i):TimeRecChart:SetWidth Trace, Width`

**Command Argument(s):** `Trace <int> {trInput=0 | trJitter=1}`  
`Width <int>`

**Example:** `:Displays:DigitizerDisplay(1):TimeRecChart:SetWidth trIn`

**Description:** Sets the width (in pixels) of the specified trace.

## Data Commands

### GetN?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:GetN? Trace

**Command Argument(s):** Trace <int> {trInput=0 | trJitter=1}

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:GetN] N

**Response Argument(s):** N <int>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:GetN? trInput  
[:Displays:DigitizerDisplay(1):TimeRecChart:GetN] 8188

**Description:** Queries the number of points in the specified trace.

### GetXArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:GetXArray? Trace, XUnit

**Command Argument(s):** Trace <int> {trInput=0 | trJitter=1}  
XUnit <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:GetXArray] XArray

**Response Argument(s):** XArray <doublearray>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:GetXArray? tr  
[:Displays:DigitizerDisplay(1):TimeRecChart:GetXArray] -  
-2.125-7, -2.00E-7, ...

**Description:** Returns the X-axis array for the specified trace in the specified units.

### GetYArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:GetYArray? Trace, YUnit

**Command Argument(s):** Trace <int> {trInput=0 | trJitter=1}  
YUnit <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:GetYArray] YArray

**Response Argument(s):** YArray <doublearray>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:GetYArray? tr  
[:Displays:DigitizerDisplay(1):TimeRecChart:GetYArray] 4

**Description:** Returns the Y-axis array for the specified trace in the specified units.

## GetXY?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:GetXY? *Trace, Index, XUnit, YUnit*

**Command Argument(s):** *Trace* <int> {trInput=0 | trJitter=1}  
*Index* <int>  
*XUnit* <string>  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:GetXY] *XYRdg*

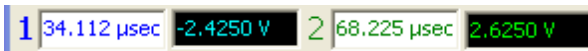
**Response Argument(s):** *XYRdg* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:GetXY? trInput  
[:Displays:DigitizerDisplay(1):TimeRecChart:GetXY] "-2.2

**Description:** Returns the X,Y pair corresponding to the Index<sup>th</sup> point of the specified trace in the specified units.

## 2.3.16.2.1.1 Digitizer Display Cursor

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): TimeRecChart:Cursor</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the cursor of the digitizer display time record graph.  Note that each Chart (TimeRec,Chart, SpecChart, ProbChart, and EyeDiagram) has its own cursor. For brevity, only the cursor of the TimeRecChart is explained.

**x1?**

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x1? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x1] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x1?

[ :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x1] 3

**Related Command(s):** x1

**Description:** Queries the cursor 1 X-value.

**x1**

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x1 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x1 2E-

**Related Command(s):** x1?

**Description:** Sets the cursor 1 X-value.

## x2?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x2? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x2] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x2?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x2] 6

**Related Command(s):** x2

**Description:** Queries the cursor 2 X-value.

## x2

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:x2 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:x2 4E-

**Related Command(s):** x2?

**Description:** Sets the cursor 2 X-value.

## y1Rdg?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:y1Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:y1Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:y1Rdg?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:y1Rdg]

**Description:** Returns the cursor 1 Y-Value.

## y2Rdg?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:y2Rdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:y2Rdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:y2Rdg?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:y2Rdg]

**Description:** Returns the cursor 2 Y-Value.



## dxCalc?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dxCalc?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dxCalc] *Value*

**Response Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dxCalc  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dxCalc]

**Related Command(s):** dxCalc

**Description:** Queries the delta-X calculation method.

## dxCalc

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dxCalc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dxCalc

**Related Command(s):** dxCalc?

**Description:** Sets the delta-X calculation method

## dxRdg?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dxRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dxRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dxRdg?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dxRdg]

**Description:** Returns the delta-X value.

## dyCalc?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dyCalc?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dyCalc] *Value*

**Response Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dyCalc  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dyCalc]

**Related Command(s):** dyCalc

**Description:** Queries the delta-Y calculation method..

## dyCalc

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dyCalc *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dyCalc

**Related Command(s):** dyCalc?

**Description:** Sets the delta-Y calculation method..

## dyRdg?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dyRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:dyRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dyRdg?  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:dyRdg]

**Description:** Queries the delta-Y value.

## GetDxUnit?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:GetDxUnit?

**Command Argument(s):** None

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:GetDxUnit] *dxUnit*

**Response Argument(s):** *dxUnit* <string>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:GetDxU  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:GetDxU]

**Description:** Returns a string containing the units of the delta-X measurement.

## GetDyUnit?

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:GetDyUnit?

**Command Argument(s):** None

**Response Syntax:** [:Displays:DigitizerDisplay(i):TimeRecChart:Cursor:GetDyUnit] *dyUnit*

**Response Argument(s):** *dyUnit* <string>

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:GetDyU  
[:Displays:DigitizerDisplay(1):TimeRecChart:Cursor:GetDy

**Description:** Returns a string containing the units of the delta-Y measurement.



## MoveToMax

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:MoveToMax *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:MoveTo

**Description:** Moves the specified cursor to the position corresponding to the maximum Y-value.

## MoveToMin

**Command Syntax:** :Displays:DigitizerDisplay(i):TimeRecChart:Cursor:MoveToMin *CursorIndex*

**Command Argument(s):** *CursorIndex* <int> {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

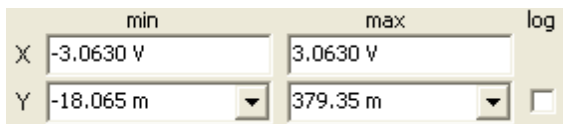
**Example:** :Displays:DigitizerDisplay(1):TimeRecChart:Cursor:MoveTo

**Description:** Moves the specified cursor to the position corresponding to the maximum Y-value.



2.3.16.2.2 Digitizer Display Probability

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): ProbChart</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Probability tab of the digitizer display.



### InputXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:InputXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:InputXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:InputXmax?  
[:Displays:DigitizerDisplay(1):ProbChart:InputXmax] 3.06

**Related Command(s):** InputXmax

**Description:** Queries the maximum X-axis value for the input trace.

### InputXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:InputXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:InputXmax *Value*

**Related Command(s):** InputXmax?

**Description:** Sets the maximum X-axis value for the input trace.

### InputXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:InputXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:InputXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:InputXmin?  
[:Displays:DigitizerDisplay(1):ProbChart:InputXmin] -3.0

**Related Command(s):** InputXmin

**Description:** Queries the minimum X-axis value for the input trace.

## InputXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:InputXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:InputXmin -3 V

**Related Command(s):** InputXmin?

**Description:** Sets the minimum X-axis value for the input trace.

## JitterXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:JitterXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:JitterXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:JitterXmax?  
 [:Displays:DigitizerDisplay(1):ProbChart:JitterXmax] 5.0

**Related Command(s):** JitterXmax

**Description:** Queries the maximum X-axis value for the jitter trace.

## JitterXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:JitterXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:JitterXmax 5.E-1

**Related Command(s):** JitterXmax?

**Description:** Sets the maximum X-axis value for the jitter trace.

## JitterXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:JitterXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:JitterXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:JitterXmin?  
 [:Displays:DigitizerDisplay(1):ProbChart:JitterXmin] -5.

**Related Command(s):** JitterXmin

**Description:** Queries the minimum X-axis value for the jitter trace.

## JitterXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:JitterXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:JitterXmin -5.00

**Related Command(s):** JitterXmin?

**Description:** Sets the minimum X-axis value for the jitter trace.

## PulseRateXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseRateXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:PulseRateXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseRateXmax?  
 [:Displays:DigitizerDisplay(1):ProbChart:PulseRateXmax]

**Related Command(s):** PulseRateXmax

**Description:** Queries the maximum X-axis value for the pulse width trace.

## PulseRateXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseRateXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseRateXmax 1.

**Related Command(s):** PulseRateXmax?

**Description:** Sets the maximum X-axis value for the pulse width trace.

## PulseRateXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseRateXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:PulseRateXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseRateXmin?  
 [:Displays:DigitizerDisplay(1):ProbChart:PulseRateXmin]

**Related Command(s):** PulseRateXmin

**Description:** Queries the minimum X-axis value for the pulse width trace.

## PulseRateXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseRateXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseRateXmin 5E

**Related Command(s):** PulseRateXmin?

**Description:** Sets the minimum X-axis value for the pulse width trace.

## PulseWidthXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmax?  
 [:Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmax]

**Related Command(s):** PulseWidthXmax

**Description:** Queries the maximum X-axis value for the pulse rate trace.

## PulseWidthXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmax 4

**Related Command(s):** PulseWidthXmax?

**Description:** Sets the maximum X-axis value for the pulse rate trace.

## PulseWidthXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmin?  
 [:Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmin]

**Related Command(s):** PulseWidthXmin

**Description:** Queries the minimum X-axis value for the pulse rate trace.

## PulseWidthXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:PulseWidthXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:PulseWidthXmin 2

**Related Command(s):** PulseWidthXmin?

**Description:** Sets the minimum X-axis value for the pulse rate trace.

## Ylog?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ylog?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:Ylog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ylog?  
[:Displays:DigitizerDisplay(1):ProbChart:Ylog] False

**Related Command(s):** Ylog

**Description:** Queries the log/linear status of the Y-axis of the probability tab of the digitizer display.

## Ylog

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ylog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ylog False

**Related Command(s):** Ylog?

**Description:** Sets the log/linear status of the Y-axis of the probability tab of the digitizer display.

## Ymax?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ymax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:Ymax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ymax?  
[:Displays:DigitizerDisplay(1):ProbChart:Ymax] 0.00379

**Related Command(s):** Ymax

**Description:** Queries the maximum Y-axis value for the probability graph.

## Ymax

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ymax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ymax 0.005

**Related Command(s):** Ymax?

**Description:** Sets the maximum Y-axis value for the probability graph.

## Ymin?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ymin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:Ymin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ymin?  
 [:Displays:DigitizerDisplay(1):ProbChart:Ymin] 0.0

**Related Command(s):** Ymin

**Description:** Queries the minimum Y-axis value for the probability graph.

## Ymin

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:Ymin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:Ymin 0.0

**Related Command(s):** Ymin?

**Description:** Sets the minimum Y-axis value for the probability graph..



## AutoScale

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:AutoScale

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):ProbChart:AutoScale

**Description:** Autoscales the X and Y axes of the probability graph.

## AutoScaleX

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):ProbChart:AutoScaleX

**Description:** Autoscales the X-axis of the probability graph.

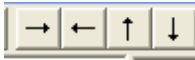
## AutoScaleY

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):ProbChart:AutoScaleY

**Description:** Autoscales the Y-axis of the probability graph.



## ShiftX

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:ShiftX *Direction*

**Command Argument(s):** *Direction* <bool> {xLeft=0|xRight=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:ShiftX xLeft

**Description:** Shifts the X-axis of the probability graph left or right.

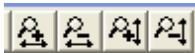
## ShiftY

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:ShiftY *Direction*

**Command Argument(s):** *Direction* <bool> {yDown=0|yUp=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:ShiftY yDown

**Description:** Shifts the Y-axis of the probability graph left or right.



## ZoomX

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:ZoomX *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmln=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:ZoomX zmOut

**Description:** Zooms the X-axis of the probability graph in or out.

## ZoomY

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:ZoomY *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmln=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:ZoomY zmOut

**Description:** Zooms the Y-axis of the probability graph in or out.



## SelectedTrace?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:SelectedTrace?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:SelectedTrace] *Value*

**Response Argument(s):** *Value* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:SelectedTrace?  
[:Displays:DigitizerDisplay(1):ProbChart:SelectedTrace]

**Related Command(s):** SelectedTrace

**Description:** Queries the selected trace of the probability graph.

## SelectedTrace

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:SelectedTrace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:SelectedTrace pI

**Related Command(s):** SelectedTrace?

**Description:** Sets the selected trace of the probability graph.

## IsTraceChecked?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:IsTraceChecked? *Trace*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:IsTraceChecked] *Checked*

**Response Argument(s):** *Checked* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:IsTraceChecked?  
[:Displays:DigitizerDisplay(1):ProbChart:IsTraceChecked]

**Description:** Queries whether the specified trace is checked (visible).

## TraceCheck

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:TraceCheck *Trace*, *Check*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*Check* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):ProbChart:TraceCheck pInput

**Description:** Checks or unchecks the specified trace of the probability graph.

## SetColor

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:SetColor *Trace*, *Color*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*Color* <int>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:SetColor pInput,

**Description:** Sets the color of the specified trace of the probability graph.



## SetWidth

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:SetWidth *Trace*, *Width*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*Width* <int>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:SetWidth pInput,

**Description:** Sets the width (in pixels) of the specified trace.

## Data Commands

### GetN?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:GetN? *Trace*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:GetN] *N*

**Response Argument(s):** *N* <int>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:GetN? pInput  
[:Displays:DigitizerDisplay(1):ProbChart:GetN] 256

**Description:** Returns the number of points in the specified trace.

### GetXArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:GetXArray? *Trace*, *XUnit*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*XUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:GetXArray] *XArray*

**Response Argument(s):** *XArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:GetXArray? pInpu  
[:Displays:DigitizerDisplay(1):ProbChart:GetXArray] "-0.  
-0.00539,-0.00534,-0.00530,....."

**Description:** Returns the X-axis array corresponding to the specified trace.

### GetYArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:GetYArray? *Trace*, *YUnit*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:GetYArray] *YArray*

**Response Argument(s):** *YArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:GetYArray? pInpu  
[:Displays:DigitizerDisplay(1):ProbChart:GetYArray] "0.0  
0,0.00134,0.0128,0.223,...."

**Description:** Returns the Y-axis array corresponding to the specified trace.

## GetXY?

**Command Syntax:** :Displays:DigitizerDisplay(i):ProbChart:GetXY? *Trace, Index, XUnit, YUnit*

**Command Argument(s):** *Trace* <int> {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}  
*Index* <int>  
*XUnit* <string>  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):ProbChart:GetXY] *XYRdg*

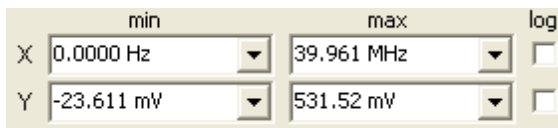
**Response Argument(s):** *XYRdg* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):ProbChart:GetXY? pPulseRate  
[:Displays:DigitizerDisplay(1):ProbChart:GetXY] "139544."

**Description:** Returns X-Y pair corresponding to the Indexth point of the specified trace in the specified units.

2.3.16.2.3 Digitizer Display Spectrum

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): SpecChart</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the spectrum tab on the digitizer display.



### InputXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXmax?

[ :Displays:DigitizerDisplay(1):SpecChart:InputXmax] 3.96

**Related Command(s):** InputXmax

**Description:** Queries the maximum X-axis value for the input spectrum.

### InputXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXmax *Value*

**Related Command(s):** InputXmax?

**Description:** Sets the maximum X-axis value for the input spectrum.

### InputXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXmin?

[ :Displays:DigitizerDisplay(1):SpecChart:InputXmin] *Value*

**Related Command(s):** InputXmin

**Description:** Queries the minimum X-axis value for the input spectrum.

## InputXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXmin *Value*

**Related Command(s):** InputXmin?

**Description:** Sets the minimum X-axis value for the input spectrum.

## InputXlog?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXlog?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputXlog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXlog?  
 [:Displays:DigitizerDisplay(1):SpecChart:InputXlog] **Fals**

**Related Command(s):** InputXlog

**Description:** Queries the log/linear status of the spectrum X-axis.

## InputXlog

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputXlog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputXlog *False*

**Related Command(s):** InputXlog?

**Description:** Sets the log/linear status of the input spectrum X-axis.

## InputYmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputYmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYmax?  
 [:Displays:DigitizerDisplay(1):SpecChart:InputYmax] **0.52**

**Related Command(s):** InputYmax

**Description:** Queries the maximum Y-axis value of the input spectrum.

## InputYmax

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYmax 0.5 V

**Related Command(s):** InputYmax?

**Description:** Sets the maximum Y-axis value of the input spectrum.

## InputYmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputYmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYmin?  
[:Displays:DigitizerDisplay(1):SpecChart:InputYmin] 0 V

**Related Command(s):** InputYmin

**Description:** Queries the minimum Y-axis value of the input spectrum.

## InputYmin

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYmin 0 V

**Related Command(s):** InputYmin?

**Description:** Sets the minimum Y-axis value of the input spectrum.

## InputYlog?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYlog?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:InputYlog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYlog?  
[:Displays:DigitizerDisplay(1):SpecChart:InputYlog] Fals

**Related Command(s):** InputYlog

**Description:** Queries the log/linear status of the input spectrum Y-axis.

## InputYlog

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:InputYlog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:InputYlog False

**Related Command(s):** InputYlog?

**Description:** Sets the log/linear status of the input spectrum Y-axis.

## JitterXmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterXmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXmax?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterXmax] 4.0

**Related Command(s):** JitterXmax

**Description:** Queries the maximum X-axis value of the jitter spectrum

## JitterXmax

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXmax 4.0E5

**Related Command(s):** JitterXmax?

**Description:** Sets the maximum X-axis value of the jitter spectrum

## JitterXmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterXmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXmin?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterXmin] 0 H

**Related Command(s):** JitterXmin

**Description:** Queries the minimum X-axis value of the jitter spectrum

## JitterXmin

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXmin 0 HZ

**Related Command(s):** JitterXmin?

**Description:** Sets the minimum X-axis value of the jitter spectrum

## JitterXlog?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXlog?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterXlog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXlog?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterXlog] Fal

**Related Command(s):** JitterXlog

**Description:** Queries the log/linear status for the jitter spectrum X-axis.

## JitterXlog

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterXlog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterXlog False

**Related Command(s):** JitterXlog?

**Description:** Sets the log/linear status of jitter spectrum X-axis.

## JitterYmax?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterYmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYmax?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterYmax] 1.1

**Related Command(s):** JitterYmax

**Description:** Queries the maximum Y-axis value for the jitter spectrum

## JitterYmax

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYmax 2.0E-

**Related Command(s):** JitterYmax?

**Description:** Sets the maximum Y-axis value for the jitter spectrum

## JitterYmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterYmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYmin?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterYmin] 9.2

**Related Command(s):** JitterYmin

**Description:** Queries the minimum Y-axis value for the jitter spectrum.

## JitterYmin

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYmin 1.0E-

**Related Command(s):** JitterYmin?

**Description:** Sets the minimum Y-axis value for the jitter spectrum.

## JitterYlog?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYlog?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:JitterYlog] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYlog?  
 [:Displays:DigitizerDisplay(1):SpecChart:JitterYlog] Fal

**Related Command(s):** JitterYlog

**Description:** Queries the log/linear status of the jitter spectrum Y-axis.



## JitterYlog

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:JitterYlog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:JitterYlog False

**Related Command(s):** JitterYlog?

**Description:** Sets the log/linear status of the jitter spectrum Y-axis.



## AutoScale

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:AutoScale

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):SpecChart:AutoScale

**Description:** Autoscales the spectrum X and Y axes.

## AutoScaleX

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):SpecChart:AutoScaleX

**Description:** Autoscales the spectrum X-axis.

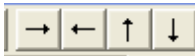
## AutoScaleY

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):SpecChart:AutoScaleY

**Description:** Autoscales the spectrum Y-axis.



## ShiftX

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:ShiftX *Direction*

**Command Argument(s):** *Direction* <bool> {xLeft=0 | xRight=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:ShiftX xLeft

**Description:** Shifts the spectrum X-axis left or right.

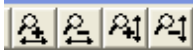
## ShiftY

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:ShiftY *Direction*

**Command Argument(s):** *Direction* <bool> {yDown=0|yUp=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:ShiftY yDown

**Description:** Shifts the spectrum Y-axis up or down.



## ZoomX

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:ZoomX *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmln=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:ZoomX zmOut

**Description:** Zooms the spectrum X-axis in or out.

## ZoomY

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:ZoomY *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmln=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:ZoomY zmOut

**Description:** Zooms the spectrum Y-axis in or out.



## SelectedTrace?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:SelectedTrace?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:SelectedTrace] *Value*

**Response Argument(s):** *Value* <int> {sInput=0|sJitter=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:SelectedTrace?

**[ :Displays:DigitizerDisplay(1):SpecChart:SelectedTrace ]**

**Related Command(s):** SelectedTrace

**Description:** Queries the selected trace of the spectrum graph.

## SelectedTrace

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:SelectedTrace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {sInput=0|sJitter=1}

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:SelectedTrace sI

**Related Command(s):** SelectedTrace?

**Description:** Sets the selected trace of the spectrum graph.

## IsTraceChecked?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:IsTraceChecked? *Trace*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:IsTraceChecked] *Checked*

**Response Argument(s):** *Checked* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:IsTraceChecked?  
[:Displays:DigitizerDisplay(1):SpecChart:IsTraceChecked]

**Description:** Queries if the specified trace is checked (visible).

## TraceCheck

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:TraceCheck *Trace, Check*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*Check* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):SpecChart:TraceCheck sInput

**Description:** Checks or unchecks the specified trace of the spectrum graph.

## SetColor

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:SetColor *Trace, Color*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*Color* <int>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:SetColor sInput,

**Description:** Sets the color of the specified trace.

## SetWidth

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:SetWidth *Trace, Width*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*Width* <int>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:SetWidth sInput,

**Description:** Sets the width (in pixels) of the specified trace.

## Data Commands

### GetN?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:GetN? *Trace*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:GetN] *N*

**Response Argument(s):** *N* <int>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:GetN? sInput  
[:Displays:DigitizerDisplay(1):SpecChart:GetN] 1024

**Description:** Returns the number of points in the specified trace of the spectrum graph..

### GetXArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:GetXArray? *Trace, XUnit*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*XUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:GetXArray] *XArray*

**Response Argument(s):** *XArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:GetXArray? sInput  
[:Displays:DigitizerDisplay(1):SpecChart:GetXArray] "0,7

**Description:** Returns the X-data array for the specified spectrum trace in the specified units.

### GetYArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:GetYArray? *Trace, YUnit*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:GetYArray] *YArray*

**Response Argument(s):** *YArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:GetYArray? sInput  
[:Displays:DigitizerDisplay(1):SpecChart:GetYArray] "4.1

**Description:** Returns the Y-data array for the specified spectrum trace in the specified units.

## GetXY?

**Command Syntax:** :Displays:DigitizerDisplay(i):SpecChart:GetXY? *Trace, Index, XUnit, YUnit*

**Command Argument(s):** *Trace* <int> {sInput=0 | sJitter=1}  
*Index* <int>  
*XUnit* <string>  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):SpecChart:GetXY] *XYRdg*

**Response Argument(s):** *XYRdg* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):SpecChart:GetXY? sInput, 1  
[:Displays:DigitizerDisplay(1):SpecChart:GetXY] "781241."

**Description:** Returns X-Y pair corresponding to the Index<sup>th</sup> point of the specified trace in the specified units.

## 2.3.16.2.4 Digitizer Display Eye Diagram

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): EyeDiagram</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the digitizer display eye diagram.

Xmin	Xmax
-39.270 nsec	353.43 nsec
Ymin	Ymax
-3.4356 V	3.4356 V

**Xmax?**

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Xmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Xmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Xmax?

[ :Displays:DigitizerDisplay(1):EyeDiagram:Xmax ] 3.5343E-

**Related Command(s):** Xmax

**Description:** Queries the maximum X-axis value of the eye diagram.

**Xmax**

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Xmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Xmax 4.0E-10 S

**Related Command(s):** Xmax?

**Description:** Sets the maximum X-axis value of the eye diagram

## Xmin?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Xmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Xmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Xmin?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Xmin] -3.9E-10

**Related Command(s):** Xmin

**Description:** Queries the minimum X-axis value of the eye diagram.

## Xmin

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Xmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Xmin -4.0E-10 S

**Related Command(s):** Xmin?

**Description:** Sets the minimum X-axis value of the eye diagram.

## Ymax?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Ymax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Ymax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Ymax?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Ymax] 3.44387

**Related Command(s):** Ymax

**Description:** Queries the maximum Y-axis value of the eye diagram.

## Ymax

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Ymax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Ymax 3 V

**Related Command(s):** Ymax?

**Description:** Sets the maximum Y-axis value of the eye diagram.

## Ymin?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Ymin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Ymin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Ymin?

[ :Displays:DigitizerDisplay(1):EyeDiagram:Ymin] -3.4 V

**Related Command(s):** Ymin

**Description:** Queries the minimum Y-axis value of the eye diagram.

## Ymin

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Ymin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Ymin -3 V

**Related Command(s):** Ymin?

**Description:** Sets the minimum Y-axis value of the eye diagram.



## AutoScale

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:AutoScale

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:AutoScale

**Description:** Autoscales the X and Y axes of the eye diagram.

## AutoScaleX

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:AutoScaleX

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:AutoScaleX

**Description:** Autoscales the X-axis of the eye diagram.

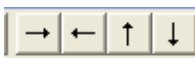
## AutoScaleY

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:AutoScaleY

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:AutoScaleY

**Description:** Autoscales the Y-axis of the eye diagram.





## ShiftX

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:ShiftX *Direction*

**Command Argument(s):** *Direction* <bool> {xLeft=0|xRight=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:ShiftX xLeft

**Description:** Shifts the X-axis of the eye diagram left or right..

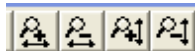
## ShiftY

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:ShiftY *Direction*

**Command Argument(s):** *Direction* <bool> {yDown=0|yUp=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:ShiftY yDown

**Description:** Shifts the Y-axis of the eye diagram left or right.



## ZoomX

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:ZoomX *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmIn=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:ZoomX zmOut

**Description:** Zooms the X-axis of the eye diagram in or out.

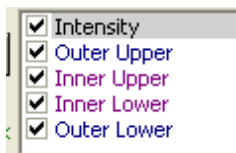
## ZoomY

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:ZoomY *Direction*

**Command Argument(s):** *Direction* <bool> {zmOut=0|zmIn=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:ZoomY zmOut

**Description:** Zooms the Y-axis of the eye diagram in or out.



## SelectedTrace?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:SelectedTrace?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:SelectedTrace] *Value*

**Response Argument(s):** *Value* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:SelectedTrace?  
[:Displays:DigitizerDisplay(1):EyeDiagram:SelectedTrace]

**Related Command(s):** SelectedTrace

**Description:** Returns the selected trace of the eye diagram.

## SelectedTrace

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:SelectedTrace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:SelectedTrace e

**Related Command(s):** SelectedTrace?

**Description:** Selects one of the eye-diagram traces.

## IsTraceChecked?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:IsTraceChecked? *Trace*

**Command Argument(s):** *Trace* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:IsTraceChecked] *Checked*

**Response Argument(s):** *Checked* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:IsTraceChecked?  
[:Displays:DigitizerDisplay(1):EyeDiagram:IsTraceChecked]

**Description:** Queries the checked (visible) status of the specified eye diagram trace.

## TraceCheck

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:TraceCheck *Trace*, *Check*

**Command Argument(s):** *Trace* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}  
*Check* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:TraceCheck edIn

**Description:** Checks or unchecks one of the eye diagram traces.

## SetColor

**Command Syntax:** `:Displays:DigitizerDisplay(i):EyeDiagram:SetColor Trace, Color`

**Command Argument(s):** `Trace <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}`  
`Color <int>`

**Example:** `:Displays:DigitizerDisplay(1):EyeDiagram:SetColor edInte`

**Description:** Sets the color of the specified trace.

## SetWidth

**Command Syntax:** `:Displays:DigitizerDisplay(i):EyeDiagram:SetWidth Trace, Width`

**Command Argument(s):** `Trace <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}`  
`Width <int>`

**Example:** `:Displays:DigitizerDisplay(1):EyeDiagram:SetWidth edInte`

**Description:** Sets the width (in pixels) of the specified trace.



## Cursor1Trace?

**Command Syntax:** `:Displays:DigitizerDisplay(i):EyeDiagram:Cursor1Trace?`

**Command Argument(s):** None

**Response Syntax:** `[:Displays:DigitizerDisplay(i):EyeDiagram:Cursor1Trace] Value`

**Response Argument(s):** `Value <int> {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}`

**Example:** `:Displays:DigitizerDisplay(1):EyeDiagram:Cursor1Trace?`

`[ :Displays:DigitizerDisplay(1):EyeDiagram:Cursor1Trace ]`

**Related Command(s):** `Cursor1Trace`

**Description:** Queries the eye diagram trace associated with cursor 1.

## Cursor1Trace

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Cursor1Trace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Cursor1Trace ed

**Related Command(s):** Cursor1Trace?

**Description:** Sets the eye diagram trace associated with cursor 1.

## Cursor2Trace?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Cursor2Trace?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Cursor2Trace] *Value*

**Response Argument(s):** *Value* <int> {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Cursor2Trace?  
 [:Displays:DigitizerDisplay(1):EyeDiagram:Cursor2Trace]

**Related Command(s):** Cursor2Trace

**Description:** Queries the eye diagram trace associated with cursor 2.

## Cursor2Trace

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Cursor2Trace *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Cursor2Trace ed

**Related Command(s):** Cursor2Trace?

**Description:** Sets the eye diagram trace associated with cursor 2.

## CursorXlock?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:CursorXlock?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:CursorXlock] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:CursorXlock?  
 [:Displays:DigitizerDisplay(1):EyeDiagram:CursorXlock] F

**Related Command(s):** CursorXlock

**Description:** Queries the on/off status of the cursor X-lock.

## CursorXlock

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:CursorXlock *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:CursorXlock Fal

**Related Command(s):** CursorXlock?

**Description:** Sets the on/off status of the cursor X-lock.



## MouseShowsZ?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:MouseShowsZ?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:MouseShowsZ] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:MouseShowsZ?  
 [:Displays:DigitizerDisplay(1):EyeDiagram:MouseShowsZ] F

**Related Command(s):** MouseShowsZ

**Description:** Queries whether a hint showing the z (probability) value will be shown when the mouse hovers over a point on the eye diagram.

## MouseShowsZ

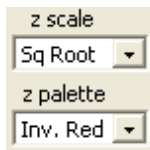
**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:MouseShowsZ *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:MouseShowsZ Fal

**Related Command(s):** MouseShowsZ?

**Description:** Sets whether a hint showing the z (probability) value will be shown when the mouse hovers over a point on the eye diagram.



## Zpalette?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Zpalette?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Zpalette] *Value*

**Response Argument(s):** *Value* <int> {Bold=0 | Spectrum=1 | Grayscale=2 | RedHot=3 | GreenHot=4 | BlueHot=5 | InvGrayscale=6 | InvRedHot=7 | InvGreenHot=8 | InvBlueHot=9 | InvSpectrum=10}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Zpalette?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Zpalette] **Bold**

**Related Command(s):** Zpalette

**Description:** Queries the eye diagram color palette.

## Zpalette

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Zpalette *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {Bold=0 | Spectrum=1 | Grayscale=2 | RedHot=3 | GreenHot=4 | BlueHot=5 | InvGrayscale=6 | InvRedHot=7 | InvGreenHot=8 | InvBlueHot=9 | InvSpectrum=10}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Zpalette **Bold**

**Related Command(s):** Zpalette?

**Description:** Sets the eye diagram color palette.

## Zscale?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Zscale?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Zscale] *Value*

**Response Argument(s):** *Value* <int> {zsLinear=0 | zsSqRoot=1 | zsLog=2}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Zscale?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Zscale] **zsLine**

**Related Command(s):** Zscale

**Description:** Queries the intensity-color mapping selection.

## Zscale

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Zscale *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {zsLinear=0 | zsSqRoot=1 | zsLog=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Zscale **zsLinear**

**Related Command(s):** Zscale?

**Description:** Sets the intensity-color mapping selection.

## Data Commands

The eye diagram data commands are divided into two groups. All the traces except intensity are represented as one-dimensional arrays with each value representing the trace data as a function of increasing time. The intensity is represented by a two-dimensional array with the X-coordinate representing time and the Y-coordinate representing voltage. The value contained at each X-Y position is the eye intensity, or probability.

### GetN?

**Command Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetN? Trace

**Command Argument(s):** Trace <int> { edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetN] N

**Response Argument(s):** N <int>

**Example:** [:Displays:DigitizerDisplay(1):EyeDiagram:GetN? edIntensi  
[:Displays:DigitizerDisplay(1):EyeDiagram:GetN] 512

**Description:** Returns the number of points in the one-dimensional data array corresponding to the specified trace.

### GetXY?

**Command Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetXY? Trace, Index, XUnit, YUnit

**Command Argument(s):** Trace <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

Index <int>

XUnit <string>

YUnit <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetXY] XYRdg

**Response Argument(s):** XYRdg <doublearray>

**Example:** [:Displays:DigitizerDisplay(1):EyeDiagram:GetXY? edIntens  
[:Displays:DigitizerDisplay(1):EyeDiagram:GetXY] "-2.39,

**Description:** Returns the X-Y pair corresponding to the Index<sup>th</sup> point in the specified one-dimensional eye diagram array.

### GetNxy?

**Command Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetNxy?

**Command Argument(s):** None

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetNxy] Nxy

**Response Argument(s):** Nxy <doublearray>

**Example:** [:Displays:DigitizerDisplay(1):EyeDiagram:GetNxy?  
[:Displays:DigitizerDisplay(1):EyeDiagram:GetNxy] "512,2

**Description:** Returns the size of the 2-dimensional array (rows, columns) corresponding to the eye diagram intensity measurement.

## GetXYZ?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:GetXYZ? *XIndex*, *YIndex*, *XUnit*, *YUnit*, *ZUnit*

**Command Argument(s):** *XIndex* <int>  
*YIndex* <int>  
*XUnit* <string>  
*YUnit* <string>  
*ZUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetXYZ] *XYZRdg*

**Response Argument(s):** *XYZRdg* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:GetXYZ? 22, 141  
 [:Displays:DigitizerDisplay(1):EyeDiagram:GetXYZ] "-2.5E

**Description:** Returns the Time,Voltage,Intensity triplet corresponding the the point at (*XIndex*, *YIndex*). (0,0) represents the lower left hand corner of the eye diagram.

## GetXArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:GetXArray? *Trace*, *XUnit*

**Command Argument(s):** *Trace* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 |  
 edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 |  
 edOutLoLimit=8}  
*XUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetXArray] *XArray*

**Response Argument(s):** *XArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:GetXArray? edIn  
 [:Displays:DigitizerDisplay(1):EyeDiagram:GetXArray] "-3

**Description:** Returns the X-axis array for the specified trace.

## GetYArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:GetYArray? *Trace*, *YUnit*

**Command Argument(s):** *Trace* <int> {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 |  
 edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 |  
 edOutLoLimit=8}  
*YUnit* <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetYArray] *YArray*

**Response Argument(s):** *YArray* <doublearray>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:GetYArray? edIn  
 [:Displays:DigitizerDisplay(1):EyeDiagram:GetYArray] "-3

**Description:** Returns the Y-axis array for the specified trace.



## GetZArray?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:GetZArray? ZUnit

**Command Argument(s):** ZUnit <string>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:GetZArray] ZArray

**Response Argument(s):** ZArray <doublearray>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:GetZArray? ""  
[:Displays:DigitizerDisplay(1):EyeDiagram:GetZArray] "0,

**Description:** Returns the complete array of intensity values for the eye diagram. The array is returned in a "raster scan" fashion with the bottom (lowest voltage) row sent first in left to right order followed by the next lowest row, etc.

## 2.3.16.2.4.1 Eye Diagram Limits

<b>Object:</b>	<b>:Displays:DigitizerDisplay(i): EyeDiagram:Limit</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the eye diagram limits.

**Limits****Exceeded?**

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Exceeded?

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:Exceeded] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Exceeded?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:Exceeded]

**Description:** Returns 1 if the currently set eye limits are exceeded, 0 if not.

 Mirror Left/Right**MirrorLR?**

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorLR?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorLR] *Value*

**Response Argument(s):** *Value* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorLR?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorLR]

**Related Command(s):** MirrorLR

**Description:** Returns the true/false status of Mirror Left-Right. When Mirror Left-Right is on, the eye limits are reflected around a vertical line in the center of the eye.

**MirrorLR**

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorLR *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value*<bool> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorLR

**Related Command(s):** MirrorLR?

**Description:** Set the true/false status of Mirror Left-Right. When Mirror Left-Right is on, the eye limits are reflected around a vertical line in the center of the eye.

 Mirror Up/Down

## MirrorUD?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorUD?

**Command Argument(s):**

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorUD] *Value*

**Response Argument(s):** *Value* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorUD?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorUD

**Related Command(s):** MirrorUD

**Description:** Returns the status of Mirror Up/Down. When Mirror Up-Down is on, the Inner Lower limit is formed by reflecting the Inner Upper Limit around the x-axis.

## MirrorUD

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:MirrorUD *Value* [, *AllowCoercion*]

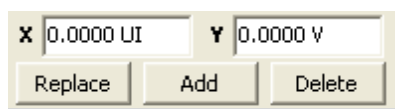
**Command Argument(s):** *Value* <bool> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:MirrorUD

**Related Command(s):** MirrorUD?

**Description:** Sets the status of Mirror Up/Down. When Mirror Up-Down is on, the Inner Lower limit is formed by reflecting the Inner Upper Limit around the x-axis..



## AddPoint?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:AddPoint? *Trace*, *X*, *Y*

**Command Argument(s):** *Trace* <int> {InUp=0 | InLo=1}

*X* <double>

*Y* <double>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:AddPoint] *Index*

**Response Argument(s):** *Index* <int>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:AddPoint?  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:AddPoint

**Description:** Adds a new point to the specified limit trace with the given X and Y value. The command returns the ordinal index of the new point..

## ChangePoint?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:ChangePoint? Trace, Index, X, Y

**Command Argument(s):** Trace <int> {InUp=0 | InLo=1}  
 Index <int>  
 X <double>  
 Y <double>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:ChangePoint] NewIndex

**Response Argument(s):** NewIndex <int>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:ChangePoint  
 [:Displays:DigitizerDisplay(1):EyeDiagram:Limit:ChangePoint]

**Description:** Changes the limit point of the specified trace with the given index to the new X and Y values. The command returns the new ordinal index of the changed point.

## DelPoint

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:DelPoint Trace, Index

**Command Argument(s):** Trace <int> {InUp=0 | InLo=1}  
 Index <int>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:DelPoint

**Description:** Deletes the point from the specified trace with the given index.

## NumPoints?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:NumPoints? Trace

**Command Argument(s):** Trace <int> {InUp=0 | InLo=1}

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:NumPoints] Count

**Response Argument(s):** Count <int>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:NumPoints  
 [:Displays:DigitizerDisplay(1):EyeDiagram:Limit:NumPoints]

**Description:** Returns the number of points in the specified limit.

## Point?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Point? Trace, Index

**Command Argument(s):** Trace <int> {InUp=0 | InLo=1}  
 Index <int>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:Point] Point

**Response Argument(s):** Point <doublearray>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Point? In  
 [:Displays:DigitizerDisplay(1):EyeDiagram:Limit:Point] "

**Description:** Returns the X,Y pair corresponding to the Index<sup>th</sup> point of the specified limit trace.

Outer Limits

Outer Upper:

Outer Lower:

## EnableOuter

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:EnableOuter On

**Command Argument(s):** On <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:EnableOut

**Description:** Enables or disables checking of the outer eye limits.

## OutLoLimit?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutLoLimit? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutLoLimit] Value

**Response Argument(s):** Value <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutLoLimi  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutLoLim

**Related Command(s):** OutLoLimit

**Description:** Queries the value of the outer lower limit.

## OutLoLimit

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutLoLimit Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutLoLimi

**Related Command(s):** OutLoLimit?

**Description:** Sets the value of the outer lower limit.

## OutUpLimit?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutUpLimit? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutUpLimit] Value

**Response Argument(s):** Value <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutUpLimi  
[:Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutUpLim

**Related Command(s):** OutUpLimit

**Description:** Queries the value of the outer upper limit.

## OutUpLimit

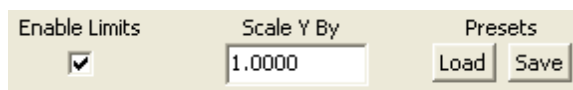
**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:OutUpLimit *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:OutUpLimit

**Related Command(s):** OutUpLimit?

**Description:** Sets the value of the outer upper limit.



## Enable

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Enable *On*

**Command Argument(s):** *On* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Enable Fa

**Description:** Enables/disables eye limit testing.

## Yscale?

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Yscale? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:DigitizerDisplay(i):EyeDiagram:Limit:Yscale] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Yscale? "  
 [:Displays:DigitizerDisplay(1):EyeDiagram:Limit:Yscale]

**Related Command(s):** Yscale

**Description:** Queries the value of the limit Y-scale factor.

## Yscale

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Yscale *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Yscale 75

**Related Command(s):** Yscale?

**Description:** Sets the value of the limit Y-scale factor.

## Load

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Load *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Load "MyE

**Description:** Loads an eye limit configuration from the named file.

## Save

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:Save *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:Save "MyE

**Description:** Saves the current eye limit configuration to the named file.

## CloseForm

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:CloseForm

**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:CloseForm

**Description:** Opens the Eye Limits form

## OpenForm

**Command Syntax:** :Displays:DigitizerDisplay(i):EyeDiagram:Limit:OpenForm

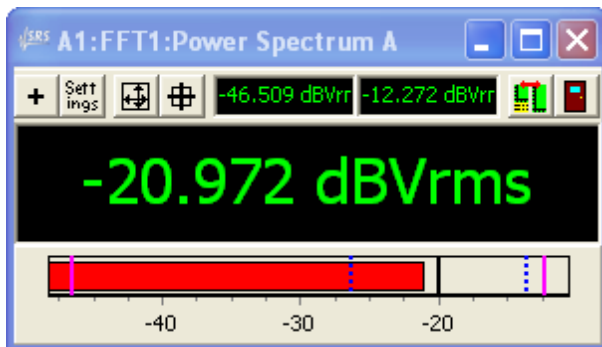
**Command Argument(s):** None

**Example:** :Displays:DigitizerDisplay(1):EyeDiagram:Limit:OpenForm

**Description:** Closes the eye limits form.

## 2.3.16.3 BarChart

<b>Object:</b>	<b>:Displays:Bar(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the bar chart.

**BarRdg?**

**Command Syntax:** :Displays:Bar(i):BarRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):BarRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):BarRdg?

[ :Displays:Bar(1):BarRdg] -20.972 DBVRMS

**Description:** Returns the current reading of the bar chart.

**AvgRdg?**

**Command Syntax:** :Displays:Bar(i):AvgRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):AvgRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):AvgRdg?

[ :Displays:Bar(1):AvgRdg] -20.112 DBVRMS

**Description:** Returns the average value of the bar chart.



## MaxRdg?

**Command Syntax:** :Displays:Bar(i):MaxRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):MaxRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):MaxRdg?

**[ :Displays:Bar(1):MaxRdg ] -19.922 DBVRMS**

**Description:** Returns the maximum bar chart reading.

## MinRdg?

**Command Syntax:** :Displays:Bar(i):MinRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):MinRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):MinRdg?

**[ :Displays:Bar(1):MinRdg ] -21.011 DBVRMS**

**Description:** Returns the minimum bar chart reading.

## SdevRdg?

**Command Syntax:** :Displays:Bar(i):SdevRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):SdevRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):SdevRdg?

**[ :Displays:Bar(1):SdevRdg ] 0.0234 VRMS**

**Description:** Queries the standard deviation of the bar chart readings.

## Reset

**Command Syntax:** :Displays:Bar(i):Reset

**Command Argument(s):** None

**Example:** :Displays:Bar(1):Reset

**Description:** Resets the calculation of maximum, minimum, and average.

## Xmax?

**Command Syntax:** :Displays:Bar(i):Xmax? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):Xmax] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):Xmax?

**[ :Displays:Bar(1):Xmax ] 1.90109E-05 VP**

**Related Command(s):** Xmax

**Description:** Queries the maximum value of the bar display.

## Xmax

**Command Syntax:** :Displays:Bar(i):Xmax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Bar(1):Xmax 2.0E-5 VP

**Related Command(s):** Xmax?

**Description:** Sets the maximum value of the bar display..

## Xmin?

**Command Syntax:** :Displays:Bar(i):Xmin? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:Displays:Bar(i):Xmin] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :Displays:Bar(1):Xmin?

**[ :Displays:Bar(1):Xmin ] 2.92475E-06 VP**

**Related Command(s):** Xmin

**Description:** Queries the minimum value of the bar display.

## Xmin

**Command Syntax:** :Displays:Bar(i):Xmin *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Bar(1):Xmin 1.E-6

**Related Command(s):** Xmin?

**Description:** Sets the minimum value of the bar display.

## Maximize?

**Command Syntax:** :Displays:Bar(i):Maximize?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):Maximize] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):Maximize?  
[:Displays:Bar(1):Maximize] False

**Related Command(s):** Maximize

**Description:** Queries the "maximized" status of the bar chart. When maximized the bar disappears and the text area expands to fill the space.

## Maximize

**Command Syntax:** :Displays:Bar(i):Maximize *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):Maximize False

**Related Command(s):** Maximize?

**Description:** Sets the "maximized" status of the bar chart. When maximized the bar disappears and the text area expands to fill the space.

## NumOnly?

**Command Syntax:** :Displays:Bar(i):NumOnly?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):NumOnly] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):NumOnly?  
[:Displays:Bar(1):NumOnly] False

**Related Command(s):** NumOnly

**Description:** Queries whether units will be displayed in the main text area of the bar chart.

## NumOnly

**Command Syntax:** :Displays:Bar(i):NumOnly *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):NumOnly False

**Related Command(s):** NumOnly?

**Description:** Sets whether units will be displayed in the main text area of the bar chart.

## SigFig?

**Command Syntax:** :Displays:Bar(i):SigFig?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):SigFig] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :Displays:Bar(1):SigFig?  
[:Displays:Bar(1):SigFig]5

**Related Command(s):** SigFig

**Description:** Queries the number of significant figures used in the bar chart display.

## SigFig

**Command Syntax:** :Displays:Bar(i):SigFig *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Bar(1):SigFig 5

**Related Command(s):** SigFig?

**Description:** Sets the number of significant figures used in the bar chart display.

## Title?

**Command Syntax:** :Displays:Bar(i):Title?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):Title] *Value*

**Response Argument(s):** *Value* <string>

**Example:** :Displays:Bar(1):Title?  
[:Displays:Bar(1):Title] A1:FFT1:Time Record A

**Related Command(s):** Title

**Description:** Returns the title string of the bar chart.

## Title

**Command Syntax:** :Displays:Bar(i):Title *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <string>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :Displays:Bar(1):Title "My Bar Chart"

**Related Command(s):** Title?

**Description:** Sets the title string of the bar chart.

## AddTrace?

**Command Syntax:** :Displays:Bar(i):AddTrace? *MeasID*

**Command Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2pectrumA=1122 | msA0FFT2pectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTpectrumA=1202 | msA0MTpectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqA=1230 | msA0MTxtalkVsFreqB=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTpectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2pectrumA=2122 | msA1FFT2pectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 | msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 | msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 | msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 | msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 | msA1MTpectrumA=2202 | msA1MTpectrumB=2203 | msA1MTthdnBinsA=2204 | msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 | msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |

```

msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Response Syntax:** [:Displays:Bar(i):AddTrace] *NewTraceID*

**Response Argument(s):** *NewTraceID* <int>

**Example:** :Displays:Bar(1):AddTrace? msA0FFTspectrum  
[:Displays:Bar(1):AddTrace] 105

**Description:** Connects the bar chart to the specified measurement. Returns the traceID of the new trace. The traceID is not needed for bar chart operation.

## GetMeasID?

**Command Syntax:** :Displays:Bar(i):GetMeasID?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Bar(i):GetMeasID] *MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTspectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |

```

msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTdA=1248 |
msA0MTdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFT2spectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTdA=2248 |
msA1MTdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :Displays:Bar(1):GetMeasID?

**[ :Displays:Bar(1):GetMeasID ]** msA0FFT2spectrum

**Description:** Returns the measurement ID currently associated with the bar chart.

## GoOffline

**Command Syntax:** :Displays:Bar(i):GoOffline

**Command Argument(s):** None

**Example:** :Displays:Bar(1):GoOffline

**Description:** Takes the bar chart offline.

## AutoScale

**Command Syntax:** :Displays:Bar(i):AutoScale

**Command Argument(s):** None

**Example:** :Displays:Bar(1):AutoScale

**Description:** Autoscales the bar portion of the display.

## Close

**Command Syntax:** :Displays:Bar(i):Close

**Command Argument(s):** None

**Example:** :Displays:Bar(1):Close

**Description:** Closes the bar chart.

## FormID?

**Command Syntax:** :Displays:Bar(i):FormID?

**Command Argument(s):** None

**Response Syntax:** [:Displays:Bar(i):FormID] FormID

**Response Argument(s):** FormID <int>

**Example:** :Displays:Bar(1):FormID?  
[:Displays:Bar(1):FormID] 25

**Description:** Returns the formID of the chart allowing the size and position to be manipulated with the standard [Form Commands](#).

## Save

**Command Syntax:** :Displays:Bar(i):Save FileName, GraphData

**Command Argument(s):** FileName <string>

GraphData <int> {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

**Example:** :Displays:Bar(1):Save "MyBar.XML", gdNeverSave

**Description:** Saves the bar chart to the specified file with the specified data options.



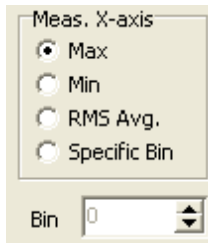
## WriteMsg

**Command Syntax:** :Displays:Bar(i):WriteMsg *Message*

**Command Argument(s):** *Message* <string>

**Example:** :Displays:Bar(1):WriteMsg Value

**Description:** Writes a user message to the bar chart text area. This command only works when the bar chart is offline.



## Mode?

**Command Syntax:** :Displays:Bar(i):Mode?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):Mode] *Value*

**Response Argument(s):** *Value* <int> {xaMax=0 | xaMin=1 | xaRMSAvg=2 | xaBin=3}

**Example:** :Displays:Bar(1):Mode?

**[ :Displays:Bar(1):Mode ] xaMax**

**Related Command(s):** Mode

**Description:** Queries the selection mode for the bar chart when the associated measurement is a vector measurement.

## Mode

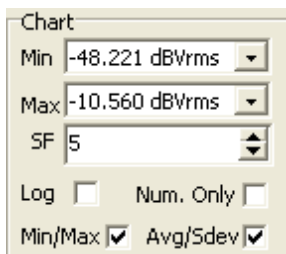
**Command Syntax:** :Displays:Bar(i):Mode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {xaMax=0 | xaMin=1 | xaRMSAvg=2 | xaBin=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):Mode xaMax

**Related Command(s):** Mode?

**Description:** Sets the selection mode for the bar chart when the associated measurement is a vector measurement.



## Readout?

**Command Syntax:** :Displays:Bar(i):Readout?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):Readout] Value

**Response Argument(s):** Value <int> {srMinMax=0 | srAvgSdev=1}

**Example:** :Displays:Bar(1):Readout?  
[:Displays:Bar(1):Readout] srMinMax

**Related Command(s):** Readout

**Description:** Queries whether the 2 smaller displays on top of the main text area show the minimum/maximum or the Average/Std. Deviation of the bar chart measurement.

## Readout

**Command Syntax:** :Displays:Bar(i):Readout Value [, AllowCoercion]

**Command Argument(s):** Value <int> {srMinMax=0 | srAvgSdev=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):Readout srMinMax

**Related Command(s):** Readout?

**Description:** Sets whether the 2 smaller displays on top of the main text area show the minimum/maximum or the Average/Std. Deviation of the bar chart measurement.

## ShowAvgSdev?

**Command Syntax:** :Displays:Bar(i):ShowAvgSdev?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):ShowAvgSdev] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):ShowAvgSdev?  
[:Displays:Bar(1):ShowAvgSdev] False

**Related Command(s):** ShowAvgSdev

**Description:** Queries whether bars corresponding to the average/standard deviation are shown in the bar area.

## ShowAvgSdev

**Command Syntax:** :Displays:Bar(i):ShowAvgSdev Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):ShowAvgSdev False

**Related Command(s):** ShowAvgSdev?

**Description:** Sets whether bars corresponding to the average/standard deviation are shown in the bar area.

## ShowMinMax?

**Command Syntax:** :Displays:Bar(i):ShowMinMax?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):ShowMinMax] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):ShowMinMax?  
[:Displays:Bar(1):ShowMinMax] **False**

**Related Command(s):** ShowMinMax

**Description:** Queries whether bars corresponding to the minimum and maximum are shown in the bar area.

## ShowMinMax

**Command Syntax:** :Displays:Bar(i):ShowMinMax *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):ShowMinMax **False**

**Related Command(s):** ShowMinMax?

**Description:** Sets whether bars corresponding to the minimum and maximum are shown in the bar area.

## Log?

**Command Syntax:** :Displays:Bar(i):Log?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):Log] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):Log?  
[:Displays:Bar(1):Log] **False**

**Related Command(s):** Log

**Description:** Queries whether the bar display is linearly or logarithmically scaled.

## Log

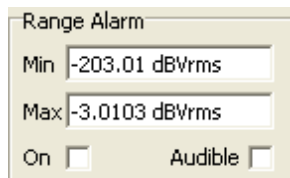
**Command Syntax:** :Displays:Bar(i):Log *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):Log **False**

**Related Command(s):** Log?

**Description:** Sets whether the bar display is linearly or logarithmically scaled.



## AlarmMax?

**Command Syntax:** `:Displays:Bar(i):AlarmMax? [ValueUnit]`

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** `[:Displays:Bar(i):AlarmMax] Value`

**Response Argument(s):** *Value* <unit>

**Example:** `:Displays:Bar(1):AlarmMax?  
[:Displays:Bar(1):AlarmMax] 1.0 VP`

**Related Command(s):** AlarmMax

**Description:** Queries the maximum value of the alarm range.

## AlarmMax

**Command Syntax:** `:Displays:Bar(i):AlarmMax Value [, AllowCoercion]`

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** `:Displays:Bar(1):AlarmMax Value`

**Related Command(s):** AlarmMax?

**Description:** Sets the maximum value of the alarm range.

## AlarmMin?

**Command Syntax:** `:Displays:Bar(i):AlarmMin? [ValueUnit]`

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** `[:Displays:Bar(i):AlarmMin] Value`

**Response Argument(s):** *Value* <unit>

**Example:** `:Displays:Bar(1):AlarmMin?  
[:Displays:Bar(1):AlarmMin] -1.0 VP`

**Related Command(s):** AlarmMin

**Description:** Queries the minimum value of the alarm range.

## AlarmMin

**Command Syntax:** :Displays:Bar(i):AlarmMin Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):AlarmMin Value

**Related Command(s):** AlarmMin?

**Description:** Sets the minimum value of the alarm range.

## RangeAlarm?

**Command Syntax:** :Displays:Bar(i):RangeAlarm?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):RangeAlarm] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):RangeAlarm?  
[:Displays:Bar(1):RangeAlarm] False

**Related Command(s):** RangeAlarm

**Description:** Queries whether to check data for range limits.

## RangeAlarm

**Command Syntax:** :Displays:Bar(i):RangeAlarm Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):RangeAlarm False

**Related Command(s):** RangeAlarm?

**Description:** Sets whether to check data for range limits. If on, the bar chart changes color, triggers an event, and possibly makes an audible sound to indicate an out-of-limits event.

## AlarmTone?

**Command Syntax:** :Displays:Bar(i):AlarmTone?

**Command Argument(s):**

**Response Syntax:** [:Displays:Bar(i):AlarmTone] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :Displays:Bar(1):AlarmTone?  
[:Displays:Bar(1):AlarmTone] False

**Related Command(s):** AlarmTone

**Description:** Queries whether an audible tone is used to signal an out-of-range condition.

## AlarmTone

**Command Syntax:** :Displays:Bar(i):AlarmTone *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :Displays:Bar(1):AlarmTone False

**Related Command(s):** AlarmTone?

**Description:** Sets whether an audible tone is used to signal an out-of-range condition.

### 2.3.17 Event Manager

<b>Object:</b>	<b>:EventMgr</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Event Manager.

<input type="checkbox"/> Analog Ch.A Scale Change	<input type="checkbox"/>	No Tone	...	No Event
<input type="checkbox"/> Analog Ch.B Scale Change	<input type="checkbox"/>	No Tone	...	No Event
<input type="checkbox"/> Analog Ch.A HiV Trip	<input type="checkbox"/>	No Tone	...	No Event
<input type="checkbox"/> Analog Ch.B HiV Trip	<input type="checkbox"/>	No Tone	...	No Event

### GetEnabled?

**Command Syntax:** :EventMgr:GetEnabled? *EventID*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

**Response Syntax:** [:EventMgr:GetEnabled] *Enabled*

**Response Argument(s):** *Enabled* <int> {False=0 | True=1}

**Example:** :EventMgr:GetEnabled? eAInARngChg  
[:EventMgr:GetEnabled] False

**Description:** Queries the enabled status for the specified event.

## SetEnabled

**Command Syntax:** :EventMgr:SetEnabled *EventID*, *Enable*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

*Enable* <bool> {False=0 | True=1}

**Example:** :EventMgr:SetEnabled eAInARngChg, False

**Description:** Sets the enabled status for the specified event.

## GetLogging?

**Command Syntax:** :EventMgr:GetLogging? *EventID*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

**Response Syntax:** [:EventMgr:GetLogging] *Logged*

**Response Argument(s):** *Logged* <int> {False=0 | True=1}

**Example:** :EventMgr:GetLogging? eAInARngChg

**[:EventMgr:GetLogging] False**

**Description:** Queries the file logging status for the specified event.



## SetLogging

**Command Syntax:** :EventMgr:SetLogging *EventID*, *Log*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

*Log* <int> {False=0 | True=1}

**Example:** :EventMgr:SetLogging eAInARngChg, False

**Description:** Sets the file logging status for the specified event.

## GetTone?

**Command Syntax:** :EventMgr:GetTone? *EventID*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

**Response Syntax:** [:EventMgr:GetTone] *Tone*

**Response Argument(s):** *Tone* <int> {tnNoTone=0 | tnTone1=1 | tnTone2=2 | tnTone3=3 | tnTone4=4 | tnTone5=5}

**Example:** :EventMgr:GetTone? eAInARngChg  
[:EventMgr:GetTone] tnNoTone

**Description:** Queries the audio tone associated with the specified event.

## SetTone

**Command Syntax:** :EventMgr:SetTone *EventID*, *Tone*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStAByte0Chg=2505 | eChStBByte0Chg=2506 | eChStAByte1Chg=2507 | eChStBByte1Chg=2508 | eChStAByte2Chg=2509 | eChStBByte2Chg=2510 | eChStAByte3Chg=2511 | eChStBByte3Chg=2512 | eChStAByte4Chg=2513 | eChStBByte4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

*Tone* <int> {tnNoTone=0 | tnTone1=1 | tnTone2=2 | tnTone3=3 | tnTone4=4 | tnTone5=5}

**Example:** :EventMgr:SetTone eAInARngChg, tnNoTone

**Description:** Sets the audio tone associated with the specified event.

## GetScript?

**Command Syntax:** :EventMgr:GetScript? *EventID*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStAByte0Chg=2505 | eChStBByte0Chg=2506 | eChStAByte1Chg=2507 | eChStBByte1Chg=2508 | eChStAByte2Chg=2509 | eChStBByte2Chg=2510 | eChStAByte3Chg=2511 | eChStBByte3Chg=2512 | eChStAByte4Chg=2513 | eChStBByte4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

**Response Syntax:** [:EventMgr:GetScript] *ScriptFile*

**Response Argument(s):** *ScriptFile* <string>

**Example:** :EventMgr:GetScript? eAInARngChg  
[:EventMgr:GetScript] "RangeScript.vbs"

**Description:** Queries the script file associated with the specified event.

## SetScript

**Command Syntax:** :EventMgr:SetScript *EventID*, *ScriptFile*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

*ScriptFile* <string>

**Example:** :EventMgr:SetScript eAInARngChg, Value

**Description:** Sets the script file associated with the specified event.

## GetComEvent?

**Command Syntax:** :EventMgr:GetComEvent? *EventID*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

**Response Syntax:** [:EventMgr:GetComEvent] *ComEvent*

**Response Argument(s):** *ComEvent* <int> {ceNoEvent=0 | ceEvent1=1 | ceEvent2=2 | ceEvent3=3 | ceEvent4=4 | ceEvent5=5 | ceSweepStart=6 | ceSweepStepStart=7 | ceSweepStepTimeout=8 | ceSweepStepDone=9 | ceSweepFinished=10 | ceBarDispLimitExceed=11 | ceGraphDispLimitExceed=12 | ceEyeLimitExceed=13 | ceKeypad=14 | ceKnob=15 | ceWarning=16 | ceCriticalError=17 | ceScriptError=18 | ceUserEvent=19}

**Example:** :EventMgr:GetComEvent? eAInARngChg  
[:EventMgr:GetComEvent] ceNoEvent

**Description:** Queries the COM event associated with the specified event.

## SetComEvent

**Command Syntax:** :EventMgr:SetComEvent *EventID*, *ComEvent*

**Command Argument(s):** *EventID* <int> {eAInARngChg=1001 | eAInBRngChg=1002 | eAInAHiV=1003 | eAInBHiV=1004 | eDInAValChg=2001 | eDInBValChg=2002 | eDInUnlockChg=2003 | eDInBiPhsChg=2004 | eDInParityChg=2005 | eChStModeChg=2501 | eChStCpChg=2502 | eChStEmphChg=2503 | eChStCRCChg=2504 | eChStABtye0Chg=2505 | eChStBBtye0Chg=2506 | eChStABtye1Chg=2507 | eChStBBtye1Chg=2508 | eChStABtye2Chg=2509 | eChStBBtye2Chg=2510 | eChStABtye3Chg=2511 | eChStBBtye3Chg=2512 | eChStABtye4Chg=2513 | eChStBBtye4Chg=2514 | eChStUserAActive=2515 | eChStUserBActive=2516 | eSwpStart=3001 | eSwpStepStart=3002 | eSwpStepTimeout=3003 | eSwpStepDone=3004 | eSwpFinished=3005 | eAlyzrA0Trig=4001 | eAlyzrA1Trig=4002 | eAlyzrNewMeas0=4003 | eAlyzrNewMeas1=4004 | eAlyzrNewMeas2=4005 | eAlyzrNewMeas3=4006 | eAlyzrNewMeas4=4007 | eDgtzrFinishedAnalysis=4008 | eBarDispLimitExceed=5001 | eGraphDispLimitExceed=5002 | eEyeLimitExceed=5003 | eInstKeypad=6001 | eInstKnob=6002 | eInstWarning=6003 | eInstCriticalError=6004 | eInstScriptTimeout=6005 | eInstScriptError=6006 | eUserEvent=6007 | eCalOutOfRng=10001}

*ComEvent* <int> {ceNoEvent=0 | ceEvent1=1 | ceEvent2=2 | ceEvent3=3 | ceEvent4=4 | ceEvent5=5 | ceSweepStart=6 | ceSweepStepStart=7 | ceSweepStepTimeout=8 | ceSweepStepDone=9 | ceSweepFinished=10 | ceBarDispLimitExceed=11 | ceGraphDispLimitExceed=12 | ceEyeLimitExceed=13 | ceKeypad=14 | ceKnob=15 | ceWarning=16 | ceCriticalError=17 | ceScriptError=18 | ceUserEvent=19}

**Example:** :EventMgr:SetComEvent eAInARngChg, ceNoEvent

**Description:** Sets the COM event associated with the specified event.

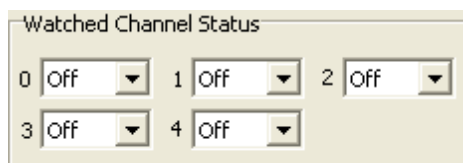
## FireUserEvent

**Command Syntax:** :EventMgr:FireUserEvent

**Command Argument(s):** None

**Example:** :EventMgr:FireUserEvent

**Description:** Fires the COM "User Event".



## GetWatchedChStat?

**Command Syntax:** :EventMgr:GetWatchedChStat? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:EventMgr:GetWatchedChStat] *Byte*

**Response Argument(s):** *Byte* <int>

**Example:** :EventMgr:GetWatchedChStat? Value  
[:EventMgr:GetWatchedChStat] Value

**Description:** Returns the channel status byte watched by the Index<sup>th</sup> (0-4) channel status watcher.

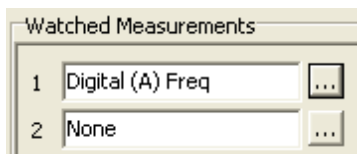
## SetWatchedChStat

**Command Syntax:** :EventMgr:SetWatchedChStat *Index, Byte*

**Command Argument(s):** *Index* <int>  
*Byte* <int>

**Example:** :EventMgr:SetWatchedChStat Value, Value

**Description:** Sets the channel status byte watched by the Index<sup>th</sup> (0-4) channel status watcher.



## GetWatchedMeas?

**Command Syntax:** :EventMgr:GetWatchedMeas? *Index*

**Command Argument(s):** *Index* <int>

**Response Syntax:** [:EventMgr:GetWatchedMeas] *MeasID*

**Response Argument(s):** *MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFT2spectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2spectrumA=1122 | msA0FFT2spectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTspectrumA=1202 | msA0MTspectrumB=1203 | msA0MTthdnBinsA=1204 |

```

msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 |
msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 |
msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 |
msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 |
msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 |
msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 |
msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 |
msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 |
msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 |
msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 |
msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 |
msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 |
msA0MTtdB=1249 | msA0MTrippleA=1250 | msA0MTrippleB=1251 |
msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 |
msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 |
msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTspectrum=2111 |
msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 |
msA1FFT2timeRecB=2121 | msA1FFT2spectrumA=2122 |
msA1FFT2spectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 |
msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 |
msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 |
msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 |
msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 |
msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 |
msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 |
msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 |
msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 |
msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 |
msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 |
msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |
msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTthdBinsA=2206 | msA1MTthdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTrippleA=2250 | msA1MTrippleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :EventMgr:GetWatchedMeas? 0

**[ :EventMgr:GetWatchedMeas ]** msA1MTimdVsFreqA

**Description:** Returns the id of the measurement watched by the Index<sup>th</sup> (0-4) measurement watcher.

## SetWatchedMeas

**Command Syntax:** :EventMgr:SetWatchedMeas *Index*, *MeasID*

**Command Argument(s):** *Index* <int>

*MeasID* <int> {msNull=0 | msAnlgFreqA=1 | msAnlgFreqB=2 | msAnlgPhase=3 | msDigFreqA=10 | msDigFreqB=11 | msDigPhase=12 | msDigCarrierFreq=14 | msDigCarrierAmp=15 | msDigCarrierDelay=16 | msA0LevelA=1100 | msA0LevelB=1101 | msA0FFTtimeRec=1110 | msA0FFTpectrum=1111 | msA0FFTlinSpec=1112 | msA0FFTlinPhase=1113 | msA0FFT2timeRecA=1120 | msA0FFT2timeRecB=1121 | msA0FFT2pectrumA=1122 | msA0FFT2pectrumB=1123 | msA0FFT2linSpecA=1124 | msA0FFT2linSpecB=1125 | msA0FFT2xferMag=1126 | msA0FFT2xferPhase=1127 | msA0FFT2coherence=1128 | msA0FFT2impulseResp=1129 | msA0TimeDomDetector=1130 | msA0FFT2anechoicRespMag=1131 | msA0FFT2anechoicRespPhase=1132 | msA0FFT2energyTimeCurve=1133 | msA0THD0=1150 | msA0THD1=1151 | msA0THDvector=1152 | msA0Imd=1160 | msA0jitFreqDomTimeRec=1170 | msA0jitFreqDomPower=1171 | msA0jitFreqDomLinSpec=1172 | msA0jitFreqDomLinPhase=1173 | msA0jitFreqDomJitter=1174 | msA0jitTimeDomJitter=1175 | msA0jitPhysSampRate=1176 | msA0HistoTimeRecA=1180 | msA0HistoTimeRecB=1181 | msA0HistoHistoA=1182 | msA0HistoHistoB=1183 | msA0HistoProbA=1184 | msA0HistoProbB=1185 | msA0HistoFitA=1186 | msA0HistoFitB=1187 | msA0HistoFitMeanA=1188 | msA0HistoFitMeanB=1189 | msA0HistoFitSigmaA=1190 | msA0HistoFitSigmaB=1191 | msA0MTtimeRecA=1200 | msA0MTtimeRecB=1201 | msA0MTpectrumA=1202 | msA0MTpectrumB=1203 | msA0MTthdnBinsA=1204 | msA0MTthdnBinsB=1205 | msA0MTthdBinsA=1206 | msA0MTthdBinsB=1207 | msA0MTimdBinsA=1208 | msA0MTimdBinsB=1209 | msA0MTnoiseBinsA=1210 | msA0MTnoiseBinsB=1211 | msA0MTtdBinsA=1212 | msA0MTtdBinsB=1213 | msA0MTfreqRespMagA=1220 | msA0MTfreqRespMagB=1221 | msA0MTfreqRespPhaseA=1222 | msA0MTfreqRespPhaseB=1223 | msA0MTthdnVsFreqA=1224 | msA0MTthdnVsFreqB=1225 | msA0MTthdVsFreqA=1226 | msA0MTthdVsFreqB=1227 | msA0MTimdVsFreqA=1228 | msA0MTimdVsFreqB=1229 | msA0MTxtalkVsFreqAB=1230 | msA0MTxtalkVsFreqBA=1231 | msA0MTthdnA=1240 | msA0MTthdnB=1241 | msA0MTthdA=1242 | msA0MTthdB=1243 | msA0MTimdA=1244 | msA0MTimdB=1245 | msA0MTnoiseA=1246 | msA0MTnoiseB=1247 | msA0MTtdA=1248 | msA0MTtdB=1249 | msA0MTripleA=1250 | msA0MTripleB=1251 | msA0MTlowestToneA=1252 | msA0MTlowestToneB=1253 | msA0MThighestToneA=1254 | msA0MThighestToneB=1255 | msA1LevelA=2100 | msA1LevelB=2101 | msA1FFTtimeRec=2110 | msA1FFTpectrum=2111 | msA1FFTlinSpec=2112 | msA1FFTlinPhase=2113 | msA1FFT2timeRecA=2120 | msA1FFT2timeRecB=2121 | msA1FFT2pectrumA=2122 | msA1FFT2pectrumB=2123 | msA1FFT2linSpecA=2124 | msA1FFT2linSpecB=2125 | msA1FFT2xferMag=2126 | msA1FFT2xferPhase=2127 | msA1FFT2coherence=2128 | msA1FFT2impulseResp=2129 | msA1TimeDomDetector=2130 | msA1FFT2anechoicRespMag=2131 | msA1FFT2anechoicRespPhase=2132 | msA1FFT2energyTimeCurve=2133 | msA1THD0=2150 | msA1THD1=2151 | msA1THDvector=2152 | msA1Imd=2160 | msA1jitFreqDomTimeRec=2170 | msA1jitFreqDomPower=2171 | msA1jitFreqDomLinSpec=2172 | msA1jitFreqDomLinPhase=2173 | msA1jitFreqDomJitter=2174 | msA1jitTimeDomJitter=2175 | msA1jitPhysSampRate=2176 | msA1HistoTimeRecA=2180 | msA1HistoTimeRecB=2181 | msA1HistoHistoA=2182 | msA1HistoHistoB=2183 | msA1HistoProbA=2184 | msA1HistoProbB=2185 | msA1HistoFitA=2186 | msA1HistoFitB=2187 | msA1HistoFitMeanA=2188 |



```

msA1HistoFitMeanB=2189 | msA1HistoFitSigmaA=2190 |
msA1HistoFitSigmaB=2191 | msA1MTtimeRecA=2200 | msA1MTtimeRecB=2201 |
msA1MTspectrumA=2202 | msA1MTspectrumB=2203 | msA1MTthdnBinsA=2204 |
msA1MTthdnBinsB=2205 | msA1MTThdBinsA=2206 | msA1MTThdBinsB=2207 |
msA1MTimdBinsA=2208 | msA1MTimdBinsB=2209 | msA1MTnoiseBinsA=2210 |
msA1MTnoiseBinsB=2211 | msA1MTtdBinsA=2212 | msA1MTtdBinsB=2213 |
msA1MTfreqRespMagA=2220 | msA1MTfreqRespMagB=2221 |
msA1MTfreqRespPhaseA=2222 | msA1MTfreqRespPhaseB=2223 |
msA1MTthdnVsFreqA=2224 | msA1MTthdnVsFreqB=2225 |
msA1MTthdVsFreqA=2226 | msA1MTthdVsFreqB=2227 |
msA1MTimdVsFreqA=2228 | msA1MTimdVsFreqB=2229 |
msA1MTxtalkVsFreqAB=2230 | msA1MTxtalkVsFreqBA=2231 |
msA1MTthdnA=2240 | msA1MTthdnB=2241 | msA1MTthdA=2242 |
msA1MTthdB=2243 | msA1MTimdA=2244 | msA1MTimdB=2245 |
msA1MTnoiseA=2246 | msA1MTnoiseB=2247 | msA1MTtdA=2248 |
msA1MTtdB=2249 | msA1MTripleA=2250 | msA1MTripleB=2251 |
msA1MTlowestToneA=2252 | msA1MTlowestToneB=2253 |
msA1MThighestToneA=2254 | msA1MThighestToneB=2255 | msSweep0=5000 |
msSweep1=5001 | msSweep2=5002 | msSweep3=5003 | msSweep4=5004 |
msSweep5=5005}

```

**Example:** :EventMgr:SetWatchedMeas 2, msA1MThighestToneB

**Description:** Sets the id of the measurement watched by the Index<sup>th</sup> (0-4) measurement watcher.

## SetLogFile

**Command Syntax:** :EventMgr:SetLogFile *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :EventMgr:SetLogFile "MyLogFile.txt"

**Description:** Sets the log file associated with the Event Manager.

## Supported Form Commands:

```

:EventMgr:OpenForm
:EventMgr:OpenFormwID?
:EventMgr:CloseForm
:EventMgr:CloseForms
:EventMgr:FormCount?
:EventMgr:FormID?

```

### 2.3.18 Switcher Configuration

<b>Object:</b>	<b>:Switcher</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to controlling the switcher network.

Physical

Name:

Connector:

Switched Ch.  Dest. Ch.

Control

Communication:

COM Port:

Chain Addr.:

### AddNewSwitch

**Command Syntax:** :Switcher:AddNewSwitch *Name*, *SwitchType*, *ControlType*, *Address*, *ChainID*, *Order*

**Command Argument(s):** *Name* <string>  
*SwitchType* <int> {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 |  
stXLRInput=3}  
*ControlType* <int> {ctSerial=0 | ctTCPIP=1}  
*Address* <string>  
*ChainID* <int>  
*Order* <int>

**Example:** To add a TCP/IP switcher:  
:Switcher:AddNewSwitch "NewBNCOut", stBNCOutput, ctTCPIP  
or to add a Serial Switcher  
:Switcher:AddNewSwitch "MySw2", 0, 0, "1", 2, 1

**Description:** Adds a new switch with the specified name and options to the switcher network. The SwitchType argument specifies the type (BNC In/Out, XLR In/Out) of the new switch. The address is a string containing the TCP/IP address:Port of the switch (the default port for the switch is 600) or a string containing the number of the COM port used for serial communication. ("1" is usually the correct string). The ChainID is the chain address set on the newly added switch. Finally, the "Order" argument specifying where the newly added switch should be placed relative to existing switches. This can affect the logical port addresses for the switches.

## ChangeDaisyChainAddress

**Command Syntax:** :Switcher:ChangeDaisyChainAddress *Type, Address, NewType, NewAddress*

**Command Argument(s):** *Type* <int> {ctSerial=0 | ctTCPIP=1}  
*Address* <string>  
*NewType* <int> {ctSerial=0 | ctTCPIP=1}  
*NewAddress* <string>

**Example:** :Switcher:ChangeDaisyChainAddress ctTCPIP, "192.168.1.15  
ctTCPIP, "192.168.1.20:600"

**Description:** Changes the communication parameters for the chain of switches at the original address to those specified by the NewType and NewAddress parameters.

## ChangeSwitchChainID

**Command Syntax:** :Switcher:ChangeSwitchChainID *Name, NewChainID*

**Command Argument(s):** *Name* <string>  
*NewChainID* <int>

**Example:** :Switcher:ChangeSwitchChainID MySwitch, 12

**Description:** Changes the ChainID for the switch with the specified name to the value specified.

## ChangeSwitchDaisyChain

**Command Syntax:** :Switcher:ChangeSwitchDaisyChain *Name, Type, Address*

**Command Argument(s):** *Name* <string>  
*Type* <int> {ctSerial=0 | ctTCPIP=1}  
*Address* <string>

**Example:** :Switcher:ChangeSwitchDaisyChain "MySwitch", ctSerial, "

**Description:** Moves the switch with the specified name from its original daisy chain to the chain specified by the Type and Address arguments.

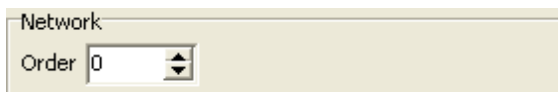
## ChangeSwitchName

**Command Syntax:** :Switcher:ChangeSwitchName *Name, NewName*

**Command Argument(s):** *Name* <string>  
*NewName* <string>

**Example:** :Switcher:ChangeSwitchName "MySwitch", "YourSwitch"

**Description:** Changes the name of the specified switch to the new value.



## ChangeSwitchOrder

**Command Syntax:** :Switcher:ChangeSwitchOrder *Name*, *NewOrder*

**Command Argument(s):** *Name* <string>  
*NewOrder* <int>

**Example:** :Switcher:ChangeSwitchOrder "MySwitch",2

**Description:** Changes the order of the named switch within its network to the new value, changing the logical port assignments.

## DeleteSwitch

**Command Syntax:** :Switcher>DeleteSwitch *Name*

**Command Argument(s):** *Name* <string>

**Example:** :Switcher>DeleteSwitch "MySwitch"

**Description:** Deletes the switch with the specified name.

## GetChainNumSwitches?

**Command Syntax:** :Switcher:GetChainNumSwitches? *ControlType*

**Command Argument(s):** *ControlType* <int> {ctSerial=0 | ctTCPIP=1}

**Response Syntax:** [:Switcher:GetChainNumSwitches] *N*

**Response Argument(s):** *N* <int>

**Example:** :Switcher:GetChainNumSwitches? ctSerial  
[:Switcher:GetChainNumSwitches] 2

**Description:** Returns the number of switches with the specified control method.

## GetChainSwitchName?

**Command Syntax:** :Switcher:GetChainSwitchName? *ControlType*, *Index*

**Command Argument(s):** *ControlType* <int> {ctSerial=0 | ctTCPIP=1}  
*Index* <int>

**Response Syntax:** [:Switcher:GetChainSwitchName] *Name*

**Response Argument(s):** *Name* <string>

**Example:** :Switcher:GetChainSwitchName? ctSerial, 0  
[:Switcher:GetChainSwitchName] MySwitch

**Description:** Returns the name of the Index<sup>th</sup> switch with the specified control method.

## GetNetNumSwitches?

**Command Syntax:** `:Switcher:GetNetNumSwitches? SwitchType`

**Command Argument(s):** `SwitchType <int> {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}`

**Response Syntax:** `[:Switcher:GetNetNumSwitches] N`

**Response Argument(s):** `N <int>`

**Example:** `:Switcher:GetNetNumSwitches? stBNCOutput  
[:Switcher:GetNetNumSwitches] 2`

**Description:** Returns the number of switches in the specified network (BNC In/Out, XLR In/Out).

## GetNetSwitchName?

**Command Syntax:** `:Switcher:GetNetSwitchName? SwitchType, Index`

**Command Argument(s):** `SwitchType <int> {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}  
Index <int>`

**Response Syntax:** `[:Switcher:GetNetSwitchName] Name`

**Response Argument(s):** `Name <string>`

**Example:** `:Switcher:GetNetSwitchName? stBNCOutput, 0  
[:Switcher:GetNetSwitchName] MySwitch`

**Description:** Returns the name of the Index<sup>th</sup> switch in the specified network (BNC In/Out, XLR In/Out).

## IdentifySwitch

**Command Syntax:** `:Switcher:IdentifySwitch Name`

**Command Argument(s):** `Name <string>`

**Example:** `:Switcher:IdentifySwitch "MySwitch"`

**Description:** Makes an audible relay-click on the specified switch.

## IsNameInUse?

**Command Syntax:** `:Switcher:IsNameInUse? Name`

**Command Argument(s):** `Name <string>`

**Response Syntax:** `[:Switcher:IsNameInUse] InUse`

**Response Argument(s):** `InUse <int>`

**Example:** `:Switcher:IsNameInUse? "MySwitch"  
[:Switcher:IsNameInUse] 1`

**Description:** Returns 1 if the specified name is already in use as a switch name, otherwise returns 0.

## MakeSwitch

**Command Syntax:** :Switcher:MakeSwitch *Name*, *PhysCh*, *State*

**Command Argument(s):** *Name* <string>  
*PhysCh* <int>  
*State* <int> {ssNone=-1 | ssA=0 | ssB=1}

**Example:** :Switcher:MakeSwitch MySwitch, 7, ssA

**Description:** Connects the physical channel (1-12) of the named switch to either A,B, or nothing (disconnected).

## MakeSwitchEnum

**Command Syntax:** :Switcher:MakeSwitchEnum *SwitchType*, *ChEnum*, *State*

**Command Argument(s):** *SwitchType* <int> {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 |  
stXLRInput=3}  
*ChEnum* <int>  
*State* <int> {ssNone=-1 | ssA=0 | ssB=1}

**Example:** :Switcher:MakeSwitchEnum stBNCOutput, 22, ssNone

**Description:** Connects the logical channel of the specified network to either A,B, or nothing (disconnected).

## ResetNetwork

**Command Syntax:** :Switcher:ResetNetwork *SwitchType*

**Command Argument(s):** *SwitchType* <int> {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 |  
stXLRInput=3}

**Example:** :Switcher:ResetNetwork stBNCOutput

**Description:** Resets (disconnects all inputs or outputs) all the switches in the specified network.

## ResetSwitch

**Command Syntax:** :Switcher:ResetSwitch *Name*

**Command Argument(s):** *Name* <string>

**Example:** :Switcher:ResetSwitch "MySwitch"

**Description:** Resets (disconnects all inputs or outputs) the named switch.

## TestSwitchComms?

**Command Syntax:** :Switcher:TestSwitchComms? *Name*

**Command Argument(s):** *Name* <string>

**Response Syntax:** [:Switcher:TestSwitchComms] *Pass*

**Response Argument(s):** *Pass* <int> {False=0 | True=1}

**Example:** :Switcher:TestSwitchComms? "MySwitch"  
[:Switcher:TestSwitchComms] True

**Description:** Returns the status of a communication test with the named switch.

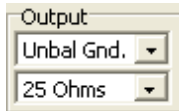
---

*Supported Form Commands:*

**:Switcher:OpenForm**  
**:Switcher:OpenFormwID?**  
**:Switcher:CloseForm**  
**:Switcher:CloseForms**  
**:Switcher:FormCount?**  
**:Switcher:FormID?**

### 2.3.19 Analog Generator

<b>Object:</b>	<b>:AnlgGen</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the analog generator.



## ConnectorConfig?

**Command Syntax:** :AnlgGen:ConnectorConfig?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:ConnectorConfig] Value

**Response Argument(s):** Value <int> {aoUnbalGnd=0 | aoUnbalFloat=1 | aoBalGnd=2 | aoBalFloat=3 | aoBalCommon=4}

**Example:** :AnlgGen:ConnectorConfig?

**[:AnlgGen:ConnectorConfig] aoUnbalGnd**

**Related Command(s):** ConnectorConfig

**Description:** Queries the analog generator connector configuration.

## ConnectorConfig

**Command Syntax:** :AnlgGen:ConnectorConfig Value [, AllowCoercion]

**Command Argument(s):** Value <int> {aoUnbalGnd=0 | aoUnbalFloat=1 | aoBalGnd=2 | aoBalFloat=3 | aoBalCommon=4}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:ConnectorConfig aoUnbalGnd

**Related Command(s):** ConnectorConfig?

**Description:** Sets the analog generator connector configuration.

## Zout?

**Command Syntax:** :AnlgGen:Zout?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Zout] Value

**Response Argument(s):** Value <int> {aozBal50Un25=0 | aozBal150Un75=1 | aoz600=2}

**Example:** :AnlgGen:Zout?

**[:AnlgGen:Zout] aozBal150Un25**

**Related Command(s):** Zout

**Description:** Queries the analog generator output impedance.



## Zout

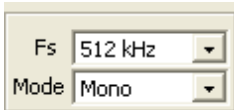
**Command Syntax:** `:AnlgGen:Zout Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {aozBa150Un25=0 | aozBa1150Un75=1 | aoz600=2}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:AnlgGen:Zout aozBa150Un25`

**Related Command(s):** Zout?

**Description:** Sets the analog generator output impedance.



## Mono?

**Command Syntax:** `:AnlgGen:Mono?`

**Command Argument(s):**

**Response Syntax:** `[:AnlgGen:Mono] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:AnlgGen:Mono?`

`[:AnlgGen:Mono] False`

**Related Command(s):** Mono

**Description:** Queries the mono/stereo mode of the analog generator.

## Mono

**Command Syntax:** `:AnlgGen:Mono Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:AnlgGen:Mono False`

**Related Command(s):** Mono?

**Description:** Sets the mono/stereo mode of the analog generator.

## SampleRate?

**Command Syntax:** `:AnlgGen:SampleRate?`

**Command Argument(s):**

**Response Syntax:** `[:AnlgGen:SampleRate] Value`

**Response Argument(s):** `Value <int> {agHz512k=0 | agHz128k=1 | agHz64k=2 | agOSR=3 | agISR=4}`

**Example:** `:AnlgGen:SampleRate?`

`[:AnlgGen:SampleRate] agHz512k`

**Related Command(s):** SampleRate

**Description:** Queries the sample rate mode of the analog generator.

## SampleRate

**Command Syntax:** :AnlgGen:SampleRate *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {agHz512k=0 | agHz128k=1 | agHz64k=2 | agOSR=3 | agISR=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:SampleRate agHz512k

**Related Command(s):** SampleRate?

**Description:** Sets the sample rate mode of the analog generator.

## SampleRateRdg?

**Command Syntax:** :AnlgGen:SampleRateRdg? [*ValueUnit*]

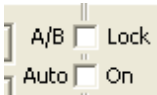
**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:SampleRateRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:SampleRateRdg?  
 [:AnlgGen:SampleRateRdg] 512000 HZ

**Description:** Queries the effective sample rate of the analog generator.



## ABLock?

**Command Syntax:** :AnlgGen:ABLock?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:ABLock] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:ABLock?  
 [:AnlgGen:ABLock] False

**Related Command(s):** ABLock

**Description:** Queries the A/B lock status. When lock is on, changes made to the channel A gain will be applied to channel B and vice versa.

## ABLock

**Command Syntax:** :AnlgGen:ABLock *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:ABLock False

**Related Command(s):** ABLock?

**Description:** Sets the A/B lock status. When lock is on, changes made to the channel A gain will be applied to channel B and vice versa.

## AutoOnOff?

**Command Syntax:** :AnlgGen:AutoOnOff?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:AutoOnOff] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:AutoOnOff?  
[:AnlgGen:AutoOnOff] **False**

**Related Command(s):** AutoOnOff

**Description:** Queries the auto-on status of the analog generator. When auto-on is active, the generator will automatically turn on at the beginning of sweeps and turn off at the end of sweeps.

## AutoOnOff

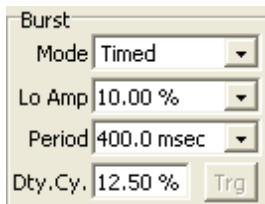
**Command Syntax:** :AnlgGen:AutoOnOff *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:AutoOnOff False

**Related Command(s):** AutoOnOff?

**Description:** Sets the auto-on status of the analog generator. When auto-on is active, the generator will automatically turn on at the beginning of sweeps and turn off at the end of sweeps.



## BurstDutyCycle?

**Command Syntax:** :AnlgGen:BurstDutyCycle? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:BurstDutyCycle] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:BurstDutyCycle?  
[:AnlgGen:BurstDutyCycle] **12.5 %**

**Related Command(s):** BurstDutyCycle

**Description:** Queries the fraction of the burst period that the generator is at its high amplitude.

## BurstDutyCycle

**Command Syntax:** :AnlgGen:BurstDutyCycle *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:BurstDutyCycle *Value*

**Related Command(s):** BurstDutyCycle?

**Description:** Sets the fraction of the burst period that the generator is at its high amplitude.

## BurstLoAmp?

**Command Syntax:** :AnlgGen:BurstLoAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:BurstLoAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:BurstLoAmp?  
 [:AnlgGen:BurstLoAmp] 10 %

**Related Command(s):** BurstLoAmp

**Description:** Queries the burst lo amplitude (as a fraction of the high amplitude).

## BurstLoAmp

**Command Syntax:** :AnlgGen:BurstLoAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:BurstLoAmp *Value*

**Related Command(s):** BurstLoAmp?

**Description:** Sets the burst lo amplitude (as a fraction of the high amplitude).

## BurstMode?

**Command Syntax:** :AnlgGen:BurstMode?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:BurstMode] *Value*

**Response Argument(s):** *Value* <int> {bmNone=0 | bmTimed=1 | bmGatedHi=2 | bmGatedLo=3 | bmShaped=4 |  
 bmTriggered=5}

**Example:** :AnlgGen:BurstMode?  
 [:AnlgGen:BurstMode] bmNone

**Related Command(s):** BurstMode

**Description:** Queries the generator burst mode.

## BurstMode

**Command Syntax:** :AnlgGen:BurstMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {bmNone=0 | bmTimed=1 | bmGatedHi=2 | bmGatedLo=3 | bmShaped=4 | bmTriggered=5}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:BurstMode bmNone

**Related Command(s):** BurstMode?

**Description:** Sets the generator burst mode.

## BurstPeriod?

**Command Syntax:** :AnlgGen:BurstPeriod? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:BurstPeriod] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:BurstPeriod?  
 [:AnlgGen:BurstPeriod] **Value**

**Related Command(s):** BurstPeriod

**Description:** Queries the burst period.

## BurstPeriod

**Command Syntax:** :AnlgGen:BurstPeriod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:BurstPeriod *Value*

**Related Command(s):** BurstPeriod?

**Description:** Sets the burst period.

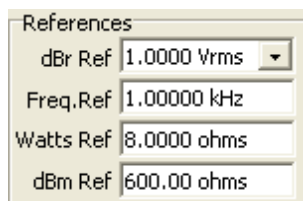
## BurstExecuteTrigger

**Command Syntax:** :AnlgGen:BurstExecuteTrigger

**Command Argument(s):** None

**Example:** :AnlgGen:BurstExecuteTrigger

**Description:** Sends a software burst trigger.



## dBmZref?

**Command Syntax:** :AnlgGen:dBmZref? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:dBmZref] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:dBmZref?  
[:AnlgGen:dBmZref] 600 OHMS

**Related Command(s):** dBmZref

**Description:** Queries the reference impedance used for dBm calculations.

## dBmZref

**Command Syntax:** :AnlgGen:dBmZref *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:dBmZref 500 OHMS

**Related Command(s):** dBmZref?

**Description:** Sets the reference impedance used for dBm calculations.

## dBrRef?

**Command Syntax:** :AnlgGen:dBrRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:dBrRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:dBrRef?  
[:AnlgGen:dBrRef] 1 VRMS

**Related Command(s):** dBrRef

**Description:** Queries the reference value used for dBr units.

## dBrRef

**Command Syntax:** :AnlgGen:dBrRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:dBrRef 1 VRMS

**Related Command(s):** dBrRef?

**Description:** Sets the reference value used for dBr units.

## FreqRef?

**Command Syntax:** :AnlgGen:FreqRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:FreqRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:FreqRef?  
[:AnlgGen:FreqRef] 1000 HZ

**Related Command(s):** FreqRef

**Description:** Queries the frequency used for relative frequency units.

## FreqRef

**Command Syntax:** :AnlgGen:FreqRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:FreqRef *Value*

**Related Command(s):** FreqRef?

**Description:** Sets the frequency used for relative frequency units.

## WattsZref?

**Command Syntax:** :AnlgGen:WattsZref? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:WattsZref] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:WattsZref?  
[:AnlgGen:WattsZref] 8 OHMS

**Related Command(s):** WattsZref

**Description:** Queries the impedance used for analog generator Watts calculation.

## WattsZref

**Command Syntax:** :AnlgGen:WattsZref *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:WattsZref *Value*

**Related Command(s):** WattsZref?

**Description:** Sets the impedance used for analog generator Watts calculation.



## SetEqFile

*Command Syntax:* :AnlgGen:SetEqFile *FileName*

*Command Argument(s):* *FileName* <string>

*Example:* :AnlgGen:SetEqFile "MyEqFile.EQ"

*Description:* Sets the generator EQ file.

## InvertEq?

*Command Syntax:* :AnlgGen:InvertEq?

*Command Argument(s):*

*Response Syntax:* [:AnlgGen:InvertEq] *Value*

*Response Argument(s):* *Value* <int> {False=0|True=1}

*Example:* :AnlgGen:InvertEq?

**[:AnlgGen:InvertEq] False**

*Related Command(s):* InvertEq

*Description:* Queries the inverted/non-inverted status of the analog generator EQ file.

## InvertEq

*Command Syntax:* :AnlgGen:InvertEq *Value* [, *AllowCoercion*]

*Command Argument(s):* *Value* <int> {False=0|True=1}

*AllowCoercion* <bool> {False=0|True=1}

*Example:* :AnlgGen:InvertEq False

*Related Command(s):* InvertEq?

*Description:* Sets the inverted/non-inverted status of the analog generator EQ file.

## [Supported Form Commands:](#)

**:AnlgGen:OpenForm**  
**:AnlgGen:OpenFormwID?**  
**:AnlgGen:CloseForm**  
**:AnlgGen:CloseForms**  
**:AnlgGen:FormCount?**  
**:AnlgGen:FormID?**



2.3.19.1 Analog Generator Channel

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>
<i>Description:</i>	Commands related to one channel of the analog generator.



### AddWaveform?

**Command Syntax:** :AnlgGen:Ch(*ch*):AddWaveform? *Type*

**Command Argument(s):** *Type* <int> {awfSine=0 | awfLoDistSine=1 | awfPhasedSine=2 | awfNoise=3 | awfUSASi=4 | awfSquare=5 | awfRamp=6 | awfArb=7 | awfChirp=8 | awfMultiTone=9 | awfIMD=10 | awfSyncBurstSine=11 | awfDC=12 | awfPolarity=13 | awfMLS=14 | awfLogSine=15}

**Response Syntax:** [:AnlgGen:Ch(*ch*):AddWaveform] *ChanID*

**Response Argument(s):** *ChanID* <int>

**Example:** :AnlgGen:Ch(0):AddWaveform? awfSine  
[:AnlgGen:Ch(0):AddWaveform] 3

**Description:** Adds a waveform of the specified type to the generator channel and returns the ID of the newly created waveform. In the example above, the properties of the new sine waveform are contained in the object:  
:AnlgGen:Ch(0):Sine(3)

### ClearWaveforms

**Command Syntax:** :AnlgGen:Ch(*ch*):ClearWaveforms

**Command Argument(s):** None

**Example:** :AnlgGen:Ch(0):ClearWaveforms

**Description:** Clears all waveforms from the generator channel.

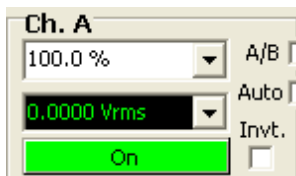
### DeleteWaveform

**Command Syntax:** :AnlgGen:Ch(*ch*):DeleteWaveform *ChanID*

**Command Argument(s):** *ChanID* <int>

**Example:** :AnlgGen:Ch(0):DeleteWaveform 3

**Description:** Clears the waveform with the specified ID.



## Gain?

**Command Syntax:** :AnlgGen:Ch(ch):Gain? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Gain] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Gain?  
[:AnlgGen:Ch(0):Gain] 100 %

**Related Command(s):** Gain

**Description:** Queries the overall gain of the generator channel.

## Gain

**Command Syntax:** :AnlgGen:Ch(ch):Gain *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Gain 50 PCT

**Related Command(s):** Gain?

**Description:** Sets the overall gain of the generator channel.

## Invert?

**Command Syntax:** :AnlgGen:Ch(ch):Invert?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Invert] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :AnlgGen:Ch(0):Invert?  
[:AnlgGen:Ch(0):Invert] *Value*

**Related Command(s):** Invert

**Description:** Queries the invert status for the generator channel.

## Invert

**Command Syntax:** :AnlgGen:Ch(ch):Invert *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Invert *Value*

**Related Command(s):** Invert?

**Description:** Sets the invert status for the generator channel.

## On?

**Command Syntax:** :AnlgGen:Ch(ch):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):On?  
 [:AnlgGen:Ch(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status for the generator channel.

## On

**Command Syntax:** :AnlgGen:Ch(ch):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status for the generator channel.

## TotalAmpRdg?

**Command Syntax:** :AnlgGen:Ch(ch):TotalAmpRdg? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):TotalAmpRdg] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):TotalAmpRdg?  
 [:AnlgGen:Ch(0):TotalAmpRdg] **1.0 VRMS**

**Description:** Queries the overall amplitude for the generator channel including the contributions of waveform amplitudes, waveform on/off, overall channel gain, and channel on/off.



## SignalCount?

**Command Syntax:** :AnlgGen:Ch(ch):SignalCount?

**Command Argument(s):** None

**Response Syntax:** [:AnlgGen:Ch(ch):SignalCount] Count

**Response Argument(s):** Count <int>

**Example:** :AnlgGen:Ch(0):SignalCount?  
[:AnlgGen:Ch(0):SignalCount] 3

**Description:** Queries the total number of waveforms active on the current channel.

## SignalID?

**Command Syntax:** :AnlgGen:Ch(ch):SignalID? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:AnlgGen:Ch(ch):SignalID] ID

**Response Argument(s):** ID <int>

**Example:** :AnlgGen:Ch(0):SignalID? 0  
[:AnlgGen:Ch(0):SignalID] 0

**Description:** Queries the ID of the Index<sup>th</sup> waveform.

## SignalType?

**Command Syntax:** :AnlgGen:Ch(ch):SignalType? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:AnlgGen:Ch(ch):SignalType] Type

**Response Argument(s):** Type <int> {awfSine=0 | awfLoDistSine=1 | awfPhasedSine=2 | awfNoise=3 | awfUSASI=4 | awfSquare=5 | awfRamp=6 | awfArb=7 | awfChirp=8 | awfMultiTone=9 | awfIMD=10 | awfSyncBurstSine=11 | awfDC=12 | awfPolarity=13 | awfMLS=14 | awfLogSine=15}

**Example:** :AnlgGen:Ch(0):SignalType? Value  
[:AnlgGen:Ch(0):SignalType] awfSine

**Description:** Queries the waveform type of the Index<sup>th</sup> waveform.

## FreqRdg?

**Command Syntax:** :AnlgGen:Ch(ch):FreqRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):FreqRdg] Value

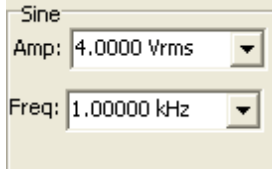
**Response Argument(s):** Value <unit>

**Example:** :AnlgGen:Ch(0):FreqRdg?  
[:AnlgGen:Ch(0):FreqRdg] 1000 HZ

**Description:** Queries the frequency of the first waveform with a defined frequency (such as sine).

2.3.19.1.1 Sine

<b>Object:</b>	<b>:AnlgGen:Ch(ch):Sine(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator sine signal.



### Amp?

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Sine(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Sine(0):Amp?

[ :AnlgGen:Ch(0):Sine(0):Amp] 4.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the sine amplitude.

### Amp

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Sine(0):Amp 4.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the sine amplitude.

### Freq?

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Sine(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Sine(0):Freq?

[ :AnlgGen:Ch(0):Sine(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the sine frequency.

## Freq

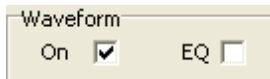
**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Sine(0):Freq 1000 HZ

**Related Command(s):** Freq?

**Description:** Sets the sine frequency.



## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Sine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Sine(0):Eq?  
 [:AnlgGen:Ch(0):Sine(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries if the generator EQ file is used to modify output amplitude. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Sine(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets if the generator EQ file is used to modify output amplitude. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq.

## On?

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Sine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Sine(0):On?  
 [:AnlgGen:Ch(0):Sine(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the sine waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Sine(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

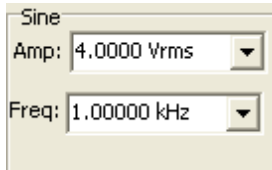
**Example:** :AnlgGen:Ch(0):Sine(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the sine waveform.

## 2.3.19.1.2 Lo Distortion Sine

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):LoDistSine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator low-distortion sine signal.



## Amp?

**Command Syntax:** :AnlgGen:Ch(*ch*):LoDistSine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):LoDistSine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Amp?

[ :AnlgGen:Ch(0):LoDistSine(0):Amp] 4.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the low-distortion sine.

## Amp

**Command Syntax:** :AnlgGen:Ch(*ch*):LoDistSine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Amp 3.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the low-distortion sine.

## Freq?

**Command Syntax:** :AnlgGen:Ch(*ch*):LoDistSine(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):LoDistSine(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Freq?

[ :AnlgGen:Ch(0):LoDistSine(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the frequency of the low-distortion sine signal.



## Freq

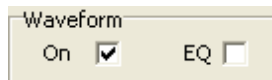
**Command Syntax:** :AnlgGen:Ch(ch):LoDistSine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Freq 1000 HZ

**Related Command(s):** Freq?

**Description:** Sets the frequency of the low-distortion sine signal.



## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):LoDistSine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):LoDistSine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Eq?  
 [:AnlgGen:Ch(0):LoDistSine(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries whether the analog generator EQ file is applied to the low-distortion sine signal.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):LoDistSine(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LoDistSine(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets whether the analog generator EQ file is applied to the low-distortion sine signal.

## On?

**Command Syntax:** :AnlgGen:Ch(ch):LoDistSine(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):LoDistSine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LoDistSine(0):On?  
 [:AnlgGen:Ch(0):LoDistSine(0):On] **False**

**Related Command(s):** On

**Description:** Queries the waveform on/off status.

## On

**Command Syntax:** :AnlgGen:Ch(ch):LoDistSine(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

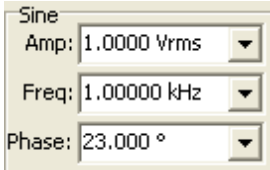
**Example:** :AnlgGen:Ch(0):LoDistSine(0):On False

**Related Command(s):** On?

**Description:** Sets the waveform on/off status.

2.3.19.1.3 Phased Sine

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):PhasedSine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator phased sine waveform.



## Amp?

**Command Syntax:** :AnlgGen:Ch(*ch*):PhasedSine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):PhasedSine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Amp?

[ :AnlgGen:Ch(0):PhasedSine(0):Amp] 1.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the phased sine waveform.

## Amp

**Command Syntax:** :AnlgGen:Ch(*ch*):PhasedSine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Amp 1.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the phased sine waveform.

## Freq?

**Command Syntax:** :AnlgGen:Ch(*ch*):PhasedSine(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):PhasedSine(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Freq?

[ :AnlgGen:Ch(0):PhasedSine(0):Freq] 10000 HZ

**Related Command(s):** Freq

**Description:** Queries the phased sine frequency.

## Freq

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Freq 100 HZ

**Related Command(s):** Freq?

**Description:** Sets the phased sine frequency.

## Phase?

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):Phase? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):PhasedSine(i):Phase] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Phase?  
 [:AnlgGen:Ch(0):PhasedSine(0):Phase] 23 °

**Related Command(s):** Phase

**Description:** Queries the phase difference between the A and B channels of the phased sine waveform.

## Phase

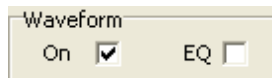
**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):Phase *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Phase 23 deg

**Related Command(s):** Phase?

**Description:** Sets the phase difference between the A and B channels of the phased sine waveform.



## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):PhasedSine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Eq?  
[:AnlgGen:Ch(0):PhasedSine(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries whether the analog generator EQ file will be applied to the phased sine waveform.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):Eq **False**

**Related Command(s):** Eq?

**Description:** Sets whether the analog generator EQ file will be applied to the phased sine waveform.

## On?

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):PhasedSine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):PhasedSine(0):On?  
[:AnlgGen:Ch(0):PhasedSine(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):PhasedSine(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

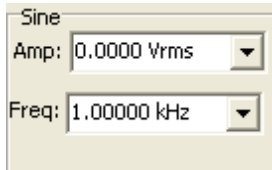
**Example:** :AnlgGen:Ch(0):PhasedSine(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

## 2.3.19.1.4 Sync Burst Sine

<b>Object:</b>	<b>:AnlgGen:Ch(ch):SyncBurstSine(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator synchronous burst sine.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Amp?

[ :AnlgGen:Ch(0):SyncBurstSine(0):Amp] 0 VRMS

**Related Command(s):** Amp

**Description:** Queries the high amplitude of the synchronous burst sine.

**Amp**

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Amp 10 VRMS

**Related Command(s):** Amp?

**Description:** Sets the high amplitude of the synchronous burst sine.

**Freq?**

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Freq?

[ :AnlgGen:Ch(0):SyncBurstSine(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the frequency of the synchronous burst sine.

## Freq

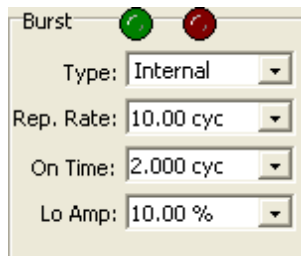
**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Freq 1000 HZ

**Related Command(s):** Freq?

**Description:** Sets the frequency of the synchronous burst sine.



## LoAmp?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):LoAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):LoAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):LoAmp?  
 [:AnlgGen:Ch(0):SyncBurstSine(0):LoAmp] 10 %

**Related Command(s):** LoAmp

**Description:** Queries the burst sine low amplitude (as a fraction of the high amplitude).

## LoAmp

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):LoAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):LoAmp *Value*

**Related Command(s):** LoAmp?

**Description:** Sets the burst sine low amplitude (as a fraction of the high amplitude).

## Mode?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Mode?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):Mode] Value

**Response Argument(s):** Value <int> {sbInternal=0 | sbExtTriggered=1 | sbExtGated=2}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Mode?

**[ :AnlgGen:Ch(0):SyncBurstSine(0):Mode] sbInternal**

**Related Command(s):** Mode

**Description:** Queries the trigger mode for the synchronous burst sine.

## Mode

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Mode Value [, AllowCoercion]

**Command Argument(s):** Value <int> {sbInternal=0 | sbExtTriggered=1 | sbExtGated=2}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Mode sbInternal

**Related Command(s):** Mode?

**Description:** Sets the trigger mode for the synchronous burst sine.

## Period?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Period? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):Period] Value

**Response Argument(s):** Value <unit>

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Period?

**[ :AnlgGen:Ch(0):SyncBurstSine(0):Period] 10 CYC**

**Related Command(s):** Period

**Description:** Queries the burst period.

## Period

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Period Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Period 10 CYC

**Related Command(s):** Period?

**Description:** Sets the burst period.



## TimeOn?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):TimeOn? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):TimeOn] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):TimeOn?  
[:AnlgGen:Ch(0):SyncBurstSine(0):TimeOn] 2 CYC

**Related Command(s):** TimeOn

**Description:** Queries the on (high-amplitude) time for the burst sine.

## TimeOn

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):TimeOn *Value* [, *AllowCoercion*]

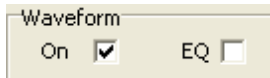
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):TimeOn 2 CYC

**Related Command(s):** TimeOn?

**Description:** Sets the on (high-amplitude) time for the burst sine.



## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Eq?  
[:AnlgGen:Ch(0):SyncBurstSine(0):Eq] False

**Related Command(s):** Eq

**Description:** Queries whether the analog generator EQ file will be applied to the burst sine waveform.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets whether the analog generator EQ file will be applied to the burst sine waveform.

## On?

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):SyncBurstSine(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):On?  
[:AnlgGen:Ch(0):SyncBurstSine(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):SyncBurstSine(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

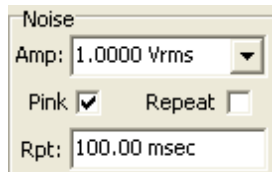
**Example:** :AnlgGen:Ch(0):SyncBurstSine(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

## 2.3.19.1.5 Noise

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):Noise(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator noise waveform.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):Noise(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):Noise(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Noise(0):Amp?

[ :AnlgGen:Ch(0):Noise(0):Amp] 1.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the noise amplitude.

**Amp**

**Command Syntax:** :AnlgGen:Ch(*ch*):Noise(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):Amp 1.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the noise amplitude.

**Pink?**

**Command Syntax:** :AnlgGen:Ch(*ch*):Noise(*i*):Pink?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(*ch*):Noise(*i*):Pink] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):Pink?

[ :AnlgGen:Ch(0):Noise(0):Pink] False

**Related Command(s):** Pink

**Description:** Queries the state of the pinking filter.

## Pink

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):Pink Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):Pink False

**Related Command(s):** Pink?

**Description:** Sets whether to use a pinking filter.

## RepeatPeriod?

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):RepeatPeriod?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Noise(i):RepeatPeriod] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):RepeatPeriod?  
[:AnlgGen:Ch(0):Noise(0):RepeatPeriod] False

**Related Command(s):** RepeatPeriod

**Description:** Queries whether the noise waveform repeats after a specified time.

## RepeatPeriod

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):RepeatPeriod Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):RepeatPeriod False

**Related Command(s):** RepeatPeriod?

**Description:** Sets whether the noise waveform repeats after a specified time.

## RepeatTime?

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):RepeatTime? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Noise(i):RepeatTime] Value

**Response Argument(s):** Value <unit>

**Example:** :AnlgGen:Ch(0):Noise(0):RepeatTime?  
[:AnlgGen:Ch(0):Noise(0):RepeatTime] 1.0 S

**Related Command(s):** RepeatTime

**Description:** Queries the repetition period when noise repeat is on.

## RepeatTime

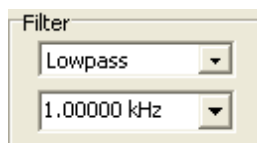
**Command Syntax:** `:AnlgGen:Ch(ch):Noise(i):RepeatTime Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:AnlgGen:Ch(0):Noise(0):RepeatTime 1.0 S`

**Related Command(s):** RepeatTime?

**Description:** Sets the repetition period when noise repeat is on.



## Filter?

**Command Syntax:** `:AnlgGen:Ch(ch):Noise(i):Filter?`

**Command Argument(s):**

**Response Syntax:** `[:AnlgGen:Ch(ch):Noise(i):Filter] Value`

**Response Argument(s):** `Value <int> {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}`

**Example:** `:AnlgGen:Ch(0):Noise(0):Filter?`  
`[:AnlgGen:Ch(0):Noise(0):Filter] nsNoFilter`

**Related Command(s):** Filter

**Description:** Queries the filter type applied to the noise.

## Filter

**Command Syntax:** `:AnlgGen:Ch(ch):Noise(i):Filter Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:AnlgGen:Ch(0):Noise(0):Filter nsNoFilter`

**Related Command(s):** Filter?

**Description:** Sets the filter type applied to the noise.

## FilterFreq?

**Command Syntax:** `:AnlgGen:Ch(ch):Noise(i):FilterFreq? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:AnlgGen:Ch(ch):Noise(i):FilterFreq] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:AnlgGen:Ch(0):Noise(0):FilterFreq?`  
`[:AnlgGen:Ch(0):Noise(0):FilterFreq] 1000 HZ`

**Related Command(s):** FilterFreq

**Description:** Queries the noise filter frequency.

## FilterFreq

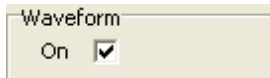
**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):FilterFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):FilterFreq 1000 HZ

**Related Command(s):** FilterFreq?

**Description:** Sets the noise filter frequency.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Noise(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):On?  
 [:AnlgGen:Ch(0):Noise(0):On] False

**Related Command(s):** On

**Description:** Queries the waveform on/off status.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

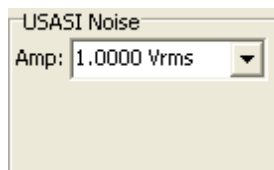
**Example:** :AnlgGen:Ch(0):Noise(0):On False

**Related Command(s):** On?

**Description:** Sets the waveform on/off status.

2.3.19.1.6 USASI Noise

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):USASI(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator USASI noise waveform.



### Amp?

**Command Syntax:** :AnlgGen:Ch(*ch*):USASI(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):USASI(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):USASI(0):Amp?  
[:AnlgGen:Ch(0):USASI(0):Amp] 1.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the USASI noise waveform.

### Amp

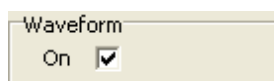
**Command Syntax:** :AnlgGen:Ch(*ch*):USASI(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):USASI(0):Amp 1.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the USASI noise waveform.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):USASI(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):USASI(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):USASI(0):On?

[ :AnlgGen:Ch(0):USASI(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):USASI(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):USASI(0):On False

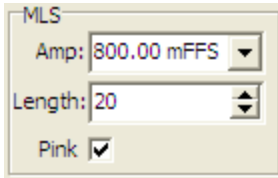
**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.



## 2.3.19.1.7 MLS Noise

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):MLS(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> as A or B, <i>i</i> as Integer
<i>Description:</i>	Commands related to the analog generator MLS noise waveform.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):MLS(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):MLS(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):MLS(0):Amp? Vrms  
[:AnlgGen:Ch(0):MLS(0):Amp] 1.0

**Related Command(s):** Amp

**Description:** Queries the sine amplitude.

**Amp**

**Command Syntax:** :AnlgGen:Ch(*ch*):MLS(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):MLS(0):Amp 1.0 Vrms

**Related Command(s):** Amp?

**Description:** Sets the sine amplitude.

## Length?

**Command Syntax:** :AnlgGen:Ch(ch):MLS(i):Length?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):MLS(i):Length] Value

**Response Argument(s):** Value <int>

**Example:** :AnlgGen:Ch(0):MLS(0):Length?  
[:AnlgGen:Ch(0):MLS(0):Length] 20

**Related Command(s):** Pink

**Description:** Queries the length of the MLS sequence.

## Length

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):Length Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):Length 20

**Related Command(s):** Pink?

**Description:** Sets the length of the MLS sequence. Sequence length is  $2^{\text{length}} - 1$  samples

## Pink?

**Command Syntax:** :AnlgGen:Ch(ch):MLS(i):Pink?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):MLS(i):Pink] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MLS(0):Pink?  
[:AnlgGen:Ch(0):MLS(0):Pink] False

**Related Command(s):** Pink

**Description:** Queries whether a pink (3 dB/oct) filter is applied to the output noise waveform.

## Pink

**Command Syntax:** :AnlgGen:Ch(ch):Noise(i):Pink Value [, AllowCoercion]

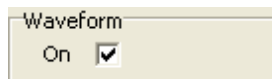
**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Noise(0):Pink False

**Related Command(s):** Pink?

**Description:** Sets whether a pink (3 dB/oct) filter is applied to the output noise waveform.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):MLS(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):MLS(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MLS(0):On?

**[ :AnlgGen:Ch(0):MLS(0):On] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the MLS waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):MLS(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MLS(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the MLS waveform.

## 2.3.19.1.8 Ramp

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):Ramp(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator ramp waveform.

Ramp

Frequency	1.00000 kHz	Frac. RiseTime	25.000 %
Low Amplitude	0.0000 Vp	High Amplitude	0.0000 Vp

**Freq?**

**Command Syntax:** :AnlgGen:Ch(*ch*):Ramp(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):Ramp(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Ramp(0):Freq?  
[:AnlgGen:Ch(0):Ramp(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the ramp frequency.

**Freq**

**Command Syntax:** :AnlgGen:Ch(*ch*):Ramp(*i*):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Ramp(0):Freq 1000

**Related Command(s):** Freq?

**Description:** Sets the ramp frequency.

**HiAmp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):Ramp(*i*):HiAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):Ramp(*i*):HiAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Ramp(0):HiAmp?  
[:AnlgGen:Ch(0):Ramp(0):HiAmp] 1.0 VP

**Related Command(s):** HiAmp

**Description:** Queries the maximum ramp amplitude.

## HiAmp

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):HiAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Ramp(0):HiAmp 1 VP

**Related Command(s):** HiAmp?

**Description:** Sets the maximum ramp amplitude.

## LoAmp?

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):LoAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Ramp(i):LoAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Ramp(0):LoAmp?  
[:AnlgGen:Ch(0):Ramp(0):LoAmp] -1.0 VP

**Related Command(s):** LoAmp

**Description:** Queries the minimum ramp amplitude.

## LoAmp

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):LoAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Ramp(0):LoAmp -1.0 VP

**Related Command(s):** LoAmp?

**Description:** Sets the minimum ramp amplitude.

## RiseFraction?

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):RiseFraction? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Ramp(i):RiseFraction] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Ramp(0):RiseFraction?  
[:AnlgGen:Ch(0):Ramp(0):RiseFraction] 25 %

**Related Command(s):** RiseFraction

**Description:** Queries the fraction of the waveform period spent going from the low amplitude to the high amplitude.

## RiseFraction

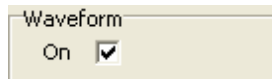
**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):RiseFraction *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Ramp(0):RiseFraction 25 PCT

**Related Command(s):** RiseFraction?

**Description:** Sets the fraction of the waveform period spent going from the low amplitude to the high amplitude.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Ramp(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Ramp(0):On?  
 [:AnlgGen:Ch(0):Ramp(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the ramp waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Ramp(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

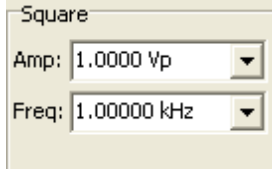
**Example:** :AnlgGen:Ch(0):Ramp(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the ramp waveform.

2.3.19.1.9 Square

<b>Object:</b>	<b>:AnlgGen:Ch(ch):Square(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator square wave.



### Amp?

**Command Syntax:** :AnlgGen:Ch(ch):Square(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Square(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Square(0):Amp?  
[:AnlgGen:Ch(0):Square(0):Amp] 1.0 VP

**Related Command(s):** Amp

**Description:** Queries the square wave amplitude.

### Amp

**Command Syntax:** :AnlgGen:Ch(ch):Square(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Square(0):Amp 1.0 VP

**Related Command(s):** Amp?

**Description:** Sets the square wave amplitude.

### Freq?

**Command Syntax:** :AnlgGen:Ch(ch):Square(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Square(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Square(0):Freq?  
[:AnlgGen:Ch(0):Square(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the square wave frequency.

## Freq

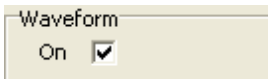
**Command Syntax:** :AnlgGen:Ch(ch):Square(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Square(0):Freq 1000 HZ

**Related Command(s):** Freq?

**Description:** Sets the square-wave frequency.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):Square(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Square(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Square(0):On?  
 [:AnlgGen:Ch(0):Square(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the square wave.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Square(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Square(0):On False

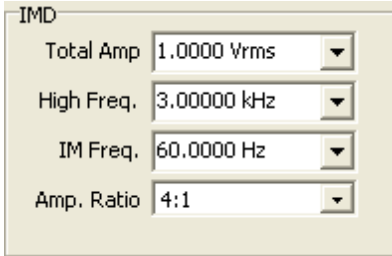
**Related Command(s):** On?

**Description:** Sets the on/off status of the square wave.



2.3.19.1.10 IMD

<b>Object:</b>	<b>:AnlgGen:Ch(ch):IMDSig(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator IMD signal.



### Amp?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):IMDSig(0):Amp?  
[:AnlgGen:Ch(0):IMDSig(0):Amp] 1.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the IMD waveform amplitude.

### Amp

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):Amp 1.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the IMD waveform amplitude.

## MainFreq?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):MainFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):MainFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):IMDSig(0):MainFreq?  
[:AnlgGen:Ch(0):IMDSig(0):MainFreq] 3000 HZ

**Related Command(s):** MainFreq

**Description:** Queries the "Main" frequency of the IMD waveform. The "Main" frequency is the High Frequency in SMPTE mode, the center frequency in CCIF mode, and the Sine Frequency in DIM mode.

## MainFreq

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):MainFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):MainFreq 3000 HZ

**Related Command(s):** MainFreq?

**Description:** Sets the "Main" frequency of the IMD waveform. The "Main" frequency is the High Frequency in SMPTE mode, the center frequency in CCIF mode, and the Sine Frequency in DIM mode.

## IMFreq?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):IMFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):IMFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):IMDSig(0):IMFreq?  
[:AnlgGen:Ch(0):IMDSig(0):IMFreq] 60 HZ

**Related Command(s):** IMFreq

**Description:** Queries the "IMD" frequency of the IMD waveform. The "IM" frequency is the Low Frequency in SMPTE mode, the difference frequency in CCIF mode, and the Square Wave Frequency in DIM mode..

## IMFreq

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):IMFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):IMFreq 60

**Related Command(s):** IMFreq?

**Description:** Sets the "IMD" frequency of the IMD waveform. The "IM" frequency is the Low Frequency in SMPTE mode, the difference frequency in CCIF mode, and the Square Wave Frequency in DIM mode..

## AmpRatio?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):AmpRatio?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):AmpRatio] *Value*

**Response Argument(s):** *Value* <int> {ar41=0|ar11=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):AmpRatio?  
 [:AnlgGen:Ch(0):IMDSig(0):AmpRatio] ar41

**Related Command(s):** AmpRatio

**Description:** Queries the tone amplitude ratio for the IMD signal.

## AmpRatio

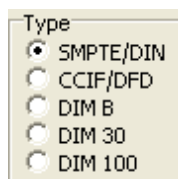
**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):AmpRatio *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {ar41=0|ar11=1}  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):AmpRatio ar41

**Related Command(s):** AmpRatio?

**Description:** Sets the tone amplitude ratio for the IMD signal. (SMPTE mode only).



## Type?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):Type?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):Type] Value

**Response Argument(s):** Value <int> {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

**Example:** :AnlgGen:Ch(0):IMDSig(0):Type?

**[ :AnlgGen:Ch(0):IMDSig(0):Type ] imdSMPTE**

**Related Command(s):** Type

**Description:** Queries the type of IMD waveform generated.

## Type

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):Type Value [, AllowCoercion]

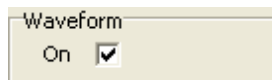
**Command Argument(s):** Value <int> {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

**AllowCoercion** <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):Type imdSMPTE

**Related Command(s):** Type?

**Description:** Sets the type of IMD waveform generated.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):IMDSig(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):IMDSig(0):On?

**[ :AnlgGen:Ch(0):IMDSig(0):On ] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the IMD waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):IMDSig(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

**AllowCoercion** <bool> {False=0 | True=1}

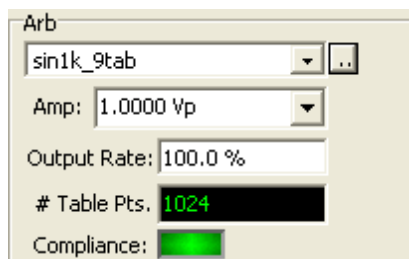
**Example:** :AnlgGen:Ch(0):IMDSig(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the IMD waveform.

2.3.19.1.11 Arbitrary

<b>Object:</b>	<b>:AnlgGen:Ch(ch):Arb(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator arbitrary waveform file.



## Amp?

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Arb(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Arb(0):Amp?  
[:AnlgGen:Ch(0):Arb(0):Amp] 1.0 VP

**Related Command(s):** Amp

**Description:** Queries the amplitude of the arbitrary waveform.

## Amp

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Arb(0):Amp 1.0 Vp

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the arbitrary waveform.

## Compliance?

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Arb(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :AnlgGen:Ch(0):Arb(0):Compliance?  
[:AnlgGen:Ch(0):Arb(0):Compliance] scOK

**Description:** Queries the compliance status of the chirp waveform.

## NumPointsRdg?

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):NumPointsRdg?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Arb(i):NumPointsRdg] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :AnlgGen:Ch(0):Arb(0):NumPointsRdg?  
[:AnlgGen:Ch(0):Arb(0):NumPointsRdg] 1024

**Description:** Queries the number of points found in the arbitrary waveform file.

## RateMultiplier?

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):RateMultiplier? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitsstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Arb(i):RateMultiplier] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Arb(0):RateMultiplier?  
[:AnlgGen:Ch(0):Arb(0):RateMultiplier] 1.0

**Related Command(s):** RateMultiplier

**Description:** Queries the output rate, in table points per output sample.

## RateMultiplier

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):RateMultiplier *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Arb(0):RateMultiplier 1.0

**Related Command(s):** RateMultiplier?

**Description:** Sets the output rate, in table points per output sample.

## Load

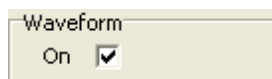
**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):Load *FileName*, *ColSelect*

**Command Argument(s):** *FileName* <string>

*ColSelect* <int>

**Example:** :AnlgGen:Ch(0):Arb(0):Load "v2.arb", 1

**Description:** Loads the data in the specified column of the specified arbitrary waveform file.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Arb(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Arb(0):On?

**[ :AnlgGen:Ch(0):Arb(0):On ] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Arb(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

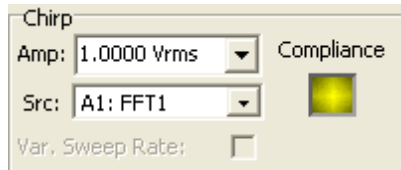
**Example:** :AnlgGen:Ch(0):Arb(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

## 2.3.19.1.12 FFT Chirp

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):Chirp(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator fft chirp waveform.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):Chirp(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):Chirp(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Chirp(0):Amp?

[ :AnlgGen:Ch(0):Chirp(0):Amp] 1.0 VRMS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the chirp waveform.

**Amp**

**Command Syntax:** :AnlgGen:Ch(*ch*):Chirp(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):Amp 1.0 VRMS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the chirp waveform.



## FFTLink?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):FFTLink?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):FFTLink] Value

**Response Argument(s):** Value <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

**Example:** :AnlgGen:Ch(0):Chirp(0):FFTLink?  
[:AnlgGen:Ch(0):Chirp(0):FFTLink] csA0FFT

**Related Command(s):** FFTLink

**Description:** Queries the linked FFT analyzer.

## FFTLink

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):FFTLink Value [, AllowCoercion]

**Command Argument(s):** Value <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):FFTLink csA0FFT

**Related Command(s):** FFTLink?

**Description:** Sets the linked FFT analyzer

## Compliance?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):Compliance] Value

**Response Argument(s):** Value <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :AnlgGen:Ch(0):Chirp(0):Compliance?  
[:AnlgGen:Ch(0):Chirp(0):Compliance] scCannotGenerate

**Description:** Queries the compliance status of the chirp waveform.

## FsMismatchRdg?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):FsMismatchRdg?

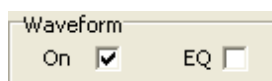
**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):FsMismatchRdg] Value

**Response Argument(s):** Value <int> {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

**Example:** :AnlgGen:Ch(0):Chirp(0):FsMismatchRdg?  
[:AnlgGen:Ch(0):Chirp(0):FsMismatchRdg] scmGenMismatched

**Description:** Queries the ratio of the generator sampling rate to the input sampling rate of the linked analyzer. If the two are not integrally related scmGenMismatched is returned.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):On?  
[:AnlgGen:Ch(0):Chirp(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):Eq?  
[:AnlgGen:Ch(0):Chirp(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries whether the current analog generator EQ file will be applied to the chirp.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):Eq **False**

**Related Command(s):** Eq?

**Description:** Sets whether the current analog generator EQ file will be applied to the chirp.

## VarSweep?

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):VarSweep?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):Chirp(i):VarSweep] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):Chirp(0):VarSweep?

**[ :AnlgGen:Ch(0):Chirp(0):VarSweep] False**

**Related Command(s):** Eq

**Description:** Queries if variable speed sweep is enabled.

## VarSweep

**Command Syntax:** :AnlgGen:Ch(ch):Chirp(i):VarSweep *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

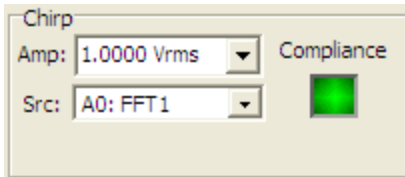
**Example:** :AnlgGen:Ch(0):Chirp(0):VarSweep False

**Related Command(s):** Eq?

**Description:** Sets if variable speed sweep is enabled. If enabled (true), the crest factor of a chirp with EQ is improved.

## 2.3.19.1.13 Log-Sine Chirp

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):LogSine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator log-sine chirp signal.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):LogSine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):LogSine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):LogSine(0):Amp?  
[:AnlgGen:Ch(0):LogSine(0):Amp] 0.8

**Related Command(s):** Amp

**Description:** Returns the log-sine chirp amplitude.

**Amp**

**Command Syntax:** :AnlgGen:Ch(*ch*):LogSine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LogSine(0):Amp 0.8

**Related Command(s):** Amp?

**Description:** Sets the log-sine chirp amplitude.

**FFTLINK?**

**Command Syntax:** :AnlgGen:Ch(*ch*):LogSine(*i*):FFTLINK?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(*ch*):LogSine(*i*):FFTLINK] *Value*

**Response Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

**Example:** :AnlgGen:Ch(0):LogSine(0):FFTLINK?  
[:AnlgGen:Ch(0):LogSine(0):FFTLINK] csA0FFT

**Related Command(s):** FFTLINK

**Description:** Returns the linked FFT analyzer.

## FFTLink

**Command Syntax:** :AnlgGen:Ch(ch):LogSine(i):FFTLink *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LogSine(0):FFTLink csA0FFT

**Related Command(s):** FFTLink?

**Description:** Sets the linked FFT analyzer

## Compliance?

**Command Syntax:** :AnlgGen:Ch(ch):LogSine(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):LogSine(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :AnlgGen:Ch(0):LogSine(0):Compliance?

**[ :AnlgGen:Ch(0):LogSine(0):Compliance ] scCannotGenerate**

**Description:** Returns the compliance status of the log-sine chirp waveform.

## FsMismatchRdg?

**Command Syntax:** :AnlgGen:Ch(ch):LogSine(i):FsMismatchRdg?

**Command Argument(s):**

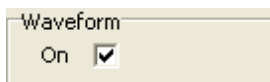
**Response Syntax:** [:AnlgGen:Ch(ch):LogSine(i):FsMismatchRdg] *Value*

**Response Argument(s):** *Value* <int> {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

**Example:** :AnlgGen:Ch(0):LogSine(0):FsMismatchRdg?

**[ :AnlgGen:Ch(0):LogSine(0):FsMismatchRdg ] scmGenMismatch**

**Description:** Queries the ratio of the generator sampling rate to the input sampling rate of the linked analyzer. If the two are not integrally related scmGenMismatched is returned.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):LogSine(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):LogSine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):LogSine(0):On?

**[ :AnlgGen:Ch(0):LogSine(0):On ] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the log-sine chirp waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):LogSine(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

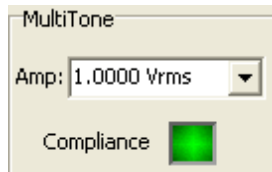
**Example:** :AnlgGen:Ch(0):LogSine(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the log-sine chirp waveform.

2.3.19.1.14 Multitone

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):MultiTone(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator multitone waveform.



### Amp?

**Command Syntax:** :AnlgGen:Ch(*ch*):MultiTone(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):MultiTone(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):MultiTone(0):Amp?  
[:AnlgGen:Ch(0):MultiTone(0):Amp] 1 VRMS

**Related Command(s):** Amp

**Description:** Queries the multitone amplitude.

### Amp

**Command Syntax:** :AnlgGen:Ch(*ch*):MultiTone(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MultiTone(0):Amp 1 VRMS

**Related Command(s):** Amp?

**Description:** Sets the multitone amplitude.

### Compliance?

**Command Syntax:** :AnlgGen:Ch(*ch*):MultiTone(*i*):Compliance?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(*ch*):MultiTone(*i*):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :AnlgGen:Ch(0):MultiTone(0):Compliance?  
[:AnlgGen:Ch(0):MultiTone(0):Compliance] scOK

**Description:** Queries the compliance status of the multitone waveform.

## ExportMultiTone

**Command Syntax:** :AnlgGen:Ch(ch):MultiTone(i):ExportMultiTone *FileName, Mode, NumBits, Dither*

**Command Argument(s):** *FileName* <string>

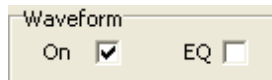
*Mode* <int> {mtArbFile=0 | mtWavFile=1}

*NumBits* <int>

*Dither* <int> {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

**Example:** :AnlgGen:Ch(0):MultiTone(0):ExportMultiTone "MyArb.arb",

**Description:** Exports the multitone waveform to either an arbitrary waveform or a .WAV file with the specified filename, bit resolution and dither options.



## Eq?

**Command Syntax:** :AnlgGen:Ch(ch):MultiTone(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):MultiTone(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MultiTone(0):Eq?

[ :AnlgGen:Ch(0):MultiTone(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries whether the current analog generator EQ file will be applied to the multitone waveform.

## Eq

**Command Syntax:** :AnlgGen:Ch(ch):MultiTone(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MultiTone(0):Eq **False**

**Related Command(s):** Eq?

**Description:** Sets whether the current analog generator EQ file will be applied to the multitone waveform.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):MultiTone(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):MultiTone(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):MultiTone(0):On?

**[ :AnlgGen:Ch(0):MultiTone(0):On ] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):MultiTone(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

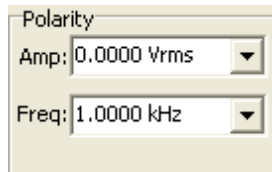
**Example:** :AnlgGen:Ch(0):MultiTone(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

## 2.3.19.1.15 Polarity

<b>Object:</b>	<b>:AnlgGen:Ch(ch):Polarity(i)</b>
<i>Object Argument(s):</i>	<i>ch &lt;int&gt;, i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the analog generator polarity waveform.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(ch):Polarity(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Polarity(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Polarity(0):Amp?

[ :AnlgGen:Ch(0):Polarity(0):Amp] 1 VRMS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the polarity waveform.

**Amp**

**Command Syntax:** :AnlgGen:Ch(ch):Polarity(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :AnlgGen:Ch(0):Polarity(0):Amp 1 VRMS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the polarity waveform.

**Freq?**

**Command Syntax:** :AnlgGen:Ch(ch):Polarity(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(ch):Polarity(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):Polarity(0):Freq?

[ :AnlgGen:Ch(0):Polarity(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the frequency of the waveform.

## Freq

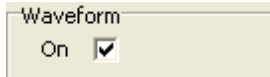
**Command Syntax:** `:AnlgGen:Ch(ch):Polarity(i):Freq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:AnlgGen:Ch(0):Polarity(0):Freq 1000 HZ`

**Related Command(s):** Freq?

**Description:** Sets the frequency of the waveform.



## On?

**Command Syntax:** `:AnlgGen:Ch(ch):Polarity(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:AnlgGen:Ch(ch):Polarity(i):On] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:AnlgGen:Ch(0):Polarity(0):On?`  
`[:AnlgGen:Ch(0):Polarity(0):On] False`

**Related Command(s):** On

**Description:** Queries the on/off status of the polarity waveform.

## On

**Command Syntax:** `:AnlgGen:Ch(ch):Polarity(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

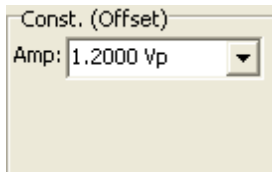
**Example:** `:AnlgGen:Ch(0):Polarity(0):On False`

**Related Command(s):** On?

**Description:** Sets the on/off status of the polarity waveform.

## 2.3.19.1.16 Constant (DC)

<b>Object:</b>	<b>:AnlgGen:Ch(<i>ch</i>):DC(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int>, <i>i</i> <int>
<i>Description:</i>	Commands related to the analog generator constant waveform.

**Amp?**

**Command Syntax:** :AnlgGen:Ch(*ch*):DC(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:AnlgGen:Ch(*ch*):DC(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :AnlgGen:Ch(0):DC(0):Amp?

[ :AnlgGen:Ch(0):DC(0):Amp] 1.2 VP

**Related Command(s):** Amp

**Description:** Queries the amplitude of the DC (constant) waveform.

**Amp**

**Command Syntax:** :AnlgGen:Ch(*ch*):DC(*i*):Amp *Value* [, *AllowCoercion*]

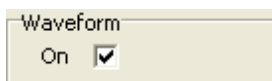
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):DC(0):Amp 1.2 VP

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the DC (constant) waveform.



## On?

**Command Syntax:** :AnlgGen:Ch(ch):DC(i):On?

**Command Argument(s):**

**Response Syntax:** [:AnlgGen:Ch(ch):DC(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):DC(0):On?  
[:AnlgGen:Ch(0):DC(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the waveform.

## On

**Command Syntax:** :AnlgGen:Ch(ch):DC(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :AnlgGen:Ch(0):DC(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the waveform.

### 2.3.20 Digital Generator

<b>Object:</b>	<b>:DigGen</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the digital generator.



## Mono?

**Command Syntax:** :DigGen:Mono?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Mono] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Mono?

**[:DigGen:Mono] False**

**Related Command(s):** Mono

**Description:** Queries the mono/stereo mode of the digital generator.

## Mono

**Command Syntax:** :DigGen:Mono Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Mono False

**Related Command(s):** Mono?

**Description:** Sets the mono/stereo mode of the digital generator.

## Dither?

**Command Syntax:** :DigGen:Dither?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Dither] Value

**Response Argument(s):** Value <int> {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

**Example:** :DigGen:Dither?

**[:DigGen:Dither] dgNoDither**

**Related Command(s):** Dither

**Description:** Queries the dither type applied to the digital audio generator.

## Dither

**Command Syntax:** `:DigGen:Dither Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Dither dgNoDither`

**Related Command(s):** Dither?

**Description:** Sets the dither type applied to the digital audio generator.



## ABlock?

**Command Syntax:** `:DigGen:ABlock?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:ABlock] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigGen:ABlock?`

`[:DigGen:ABlock] False`

**Related Command(s):** ABlock

**Description:** Queries the A/B lock status for the digital audio generator. When A/B lock is on, A-channel gain settings will be applied to the B-channel as well, and vice-versa.

## ABlock

**Command Syntax:** `:DigGen:ABlock Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:ABlock False`

**Related Command(s):** ABlock?

**Description:** Sets the A/B lock status for the digital audio generator. When A/B lock is on, A-channel gain settings will be applied to the B-channel as well, and vice-versa.

## AutoOnOff?

**Command Syntax:** :DigGen:AutoOnOff?

**Command Argument(s):**

**Response Syntax:** [:DigGen:AutoOnOff] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:AutoOnOff?  
[:DigGen:AutoOnOff] False

**Related Command(s):** AutoOnOff

**Description:** Queries the auto-on status for the digital audio generator. When auto-on is on, the generator will turn on at the beginning of sweeps and turn off when the sweep is completed.

## AutoOnOff

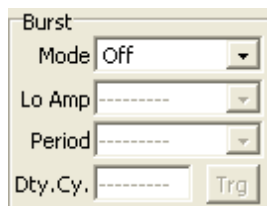
**Command Syntax:** :DigGen:AutoOnOff Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:AutoOnOff False

**Related Command(s):** AutoOnOff?

**Description:** Sets the auto-on status for the digital audio generator. When auto-on is on, the generator will turn on at the beginning of sweeps and turn off when the sweep is completed.



## BurstDutyCycle?

**Command Syntax:** :DigGen:BurstDutyCycle? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:DigGen:BurstDutyCycle] Value

**Response Argument(s):** Value <unit>

**Example:** :DigGen:BurstDutyCycle?  
[:DigGen:BurstDutyCycle] 25 PCT

**Related Command(s):** BurstDutyCycle

**Description:** Queries the fraction of the burst period that the output is in its high-amplitude state.



## BurstDutyCycle

**Command Syntax:** :DigGen:BurstDutyCycle *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:BurstDutyCycle *Value*

**Related Command(s):** BurstDutyCycle?

**Description:** Sets the fraction of the burst period that the output is in its high-amplitude state

## BurstLoAmp?

**Command Syntax:** :DigGen:BurstLoAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:BurstLoAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:BurstLoAmp?  
 [:DigGen:BurstLoAmp] -20.0 DB

**Related Command(s):** BurstLoAmp

**Description:** Queries the value of the burst "low-amplitude." The value is given as a fraction of the high amplitude.

## BurstLoAmp

**Command Syntax:** :DigGen:BurstLoAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:BurstLoAmp *Value*

**Related Command(s):** BurstLoAmp?

**Description:** Sets the value of the burst "low-amplitude." The value is given as a fraction of the high amplitude.

## BurstMode?

**Command Syntax:** :DigGen:BurstMode?

**Command Argument(s):**

**Response Syntax:** [:DigGen:BurstMode] *Value*

**Response Argument(s):** *Value* <int> {bmNone=0 | bmTimed=1 | bmGatedHi=2 | bmGatedLo=3 | bmShaped=4 | bmTriggered=5}

**Example:** :DigGen:BurstMode?  
 [:DigGen:BurstMode] bmNone

**Related Command(s):** BurstMode

**Description:** Queries the type of burst for the digital generator.

## BurstMode

**Command Syntax:** :DigGen:BurstMode *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {bmNone=0 | bmTimed=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:BurstMode bmNone

**Related Command(s):** BurstMode?

**Description:** Sets the type of burst for the digital generator.

## BurstPeriod?

**Command Syntax:** :DigGen:BurstPeriod? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:BurstPeriod] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:BurstPeriod?  
 [:DigGen:BurstPeriod] 1.0 S

**Related Command(s):** BurstPeriod

**Description:** Queries the burst period.

## BurstPeriod

**Command Syntax:** :DigGen:BurstPeriod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:BurstPeriod *Value*

**Related Command(s):** BurstPeriod?

**Description:** Sets the burst period.

References	
dBr Ref	1.0000 FFS
Freq.Ref	1.00000 kHz
V FS	1.0000 Vrms

## dBrRef?

**Command Syntax:** :DigGen:dBrRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:dBrRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:dBrRef?  
[:DigGen:dBrRef] 0.1 FFS

**Related Command(s):** dBrRef

**Description:** Queries the reference value for digital generator dBr units.

## dBrRef

**Command Syntax:** :DigGen:dBrRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:dBrRef 0.1 FFS

**Related Command(s):** dBrRef?

**Description:** Sets the reference value for digital generator dBr units.

## FreqRef?

**Command Syntax:** :DigGen:FreqRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:FreqRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:FreqRef?  
[:DigGen:FreqRef] 1000 HZ

**Related Command(s):** FreqRef

**Description:** Queries the reference frequency for digital generator relative frequency units.

## FreqRef

**Command Syntax:** :DigGen:FreqRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:FreqRef 1000

**Related Command(s):** FreqRef?

**Description:** Sets the reference frequency for digital generator relative frequency units.

## VfsRef?

**Command Syntax:** :DigGen:VfsRef? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:VfsRef] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:VfsRef?  
[:DigGen:VfsRef] 5 VRMS

**Related Command(s):** VfsRef

**Description:** Queries the "Volts Full Scale" reference.

## VfsRef

**Command Syntax:** :DigGen:VfsRef *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:VfsRef 5 VRMS

**Related Command(s):** VfsRef?

**Description:** Sets the "Volts Full Scale" reference.



## SetEqFile

**Command Syntax:** :DigGen:SetEqFile *FileName*

**Command Argument(s):** *FileName* <string>

**Example:** :DigGen:SetEqFile "MyEQ.eq"

**Description:** Sets the EQ file for the digital generator.

## InvertEq?

**Command Syntax:** :DigGen:InvertEq?

**Command Argument(s):**

**Response Syntax:** [:DigGen:InvertEq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:InvertEq?  
[:DigGen:InvertEq] False

**Related Command(s):** InvertEq

**Description:** Queries the invert/non-invert status for the digital generator EQ file.

## InvertEq

**Command Syntax:** :DigGen:InvertEq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:InvertEq False

**Related Command(s):** InvertEq?

**Description:** Sets the invert/non-invert status for the digital generator EQ file.

## [Supported Form Commands:](#)

**:DigGen:OpenForm**

**:DigGen:OpenFormwID?**

**:DigGen:CloseForm**

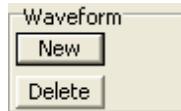
**:DigGen:CloseForms**

**:DigGen:FormCount?**

**:DigGen:FormID?**

## 2.3.20.1 Digital Generator Channel

<b>Object:</b>	<b>:DigGen:Ch(ch)</b>
<i>Object Argument(s):</i>	<i>ch</i> <int> {0=chA, 1=chB}
<i>Description:</i>	Commands associated with one channel of the digital audio generator.



## AddWaveform?

**Command Syntax:** :DigGen:Ch(ch):AddWaveform? *Type*

**Command Argument(s):** *Type* <int> {dwfSine=0 | dwfPhasedSine=1 | dwfNoise=2 | dwfUSASi=3 | dwfSquare=4 | dwfRamp=5 | dwfArb=6 | dwfChirp=7 | dwfMultiTone=8 | dwfIMD=9 | dwfConstant=10 | dwfCount=11 | dwfRotate=12 | dwfStair=13 | dwfJTest=14 | dwfPolarity=15 | dwfMLS=16 | dwfLogSine=17}

**Response Syntax:** [:DigGen:Ch(ch):AddWaveform] *ChanID*

**Response Argument(s):** *ChanID* <int>

**Example:** :DigGen:Ch(0):AddWaveform? dwfSine  
[:DigGen:Ch(0):AddWaveform] 3

**Description:** Adds a waveform of the specified type to the digital generator. The value returned is the ID of the new waveform. In the example shown above the properties of the new sine waveform could be manipulated using the object:  
:DigGen:Ch(0):Sine(3)

## ClearWaveforms

**Command Syntax:** :DigGen:Ch(ch):ClearWaveforms

**Command Argument(s):** None

**Example:** :DigGen:Ch(0):ClearWaveforms

**Description:** Clears all waveforms from the digital generator.

## DeleteWaveform

**Command Syntax:** :DigGen:Ch(ch):DeleteWaveform *ChanID*

**Command Argument(s):** *ChanID* <int>

**Example:** :DigGen:Ch(0):DeleteWaveform Value

**Description:** Deletes the the waveform with the specified id.

## SignalCount?

**Command Syntax:** :DigGen:Ch(ch):SignalCount?

**Command Argument(s):** None

**Response Syntax:** [:DigGen:Ch(ch):SignalCount] Count

**Response Argument(s):** Count <int>

**Example:** :DigGen:Ch(0):SignalCount?  
[:DigGen:Ch(0):SignalCount] 4

**Description:** Returns the current total number of waveforms configured in the digital audio generator.

## SignalID?

**Command Syntax:** :DigGen:Ch(ch):SignalID? Index

**Command Argument(s):** Index <int>

**Response Syntax:** [:DigGen:Ch(ch):SignalID] ID

**Response Argument(s):** ID <int>

**Example:** :DigGen:Ch(0):SignalID? 2  
[:DigGen:Ch(0):SignalID] 4

**Description:** Returns the ID of the Index<sup>th</sup> waveform.

## SignalType?

**Command Syntax:** :DigGen:Ch(ch):SignalType? Index

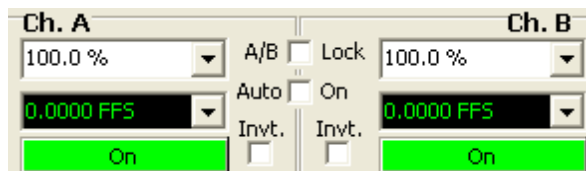
**Command Argument(s):** Index <int>

**Response Syntax:** [:DigGen:Ch(ch):SignalType] Type

**Response Argument(s):** Type <int> {dwfSine=0 | dwfPhasedSine=1 | dwfNoise=2 | dwfUSASI=3 | dwfSquare=4 | dwfRamp=5 | dwfArb=6 | dwfChirp=7 | dwfMultiTone=8 | dwfIMD=9 | dwfConstant=10 | dwfCount=11 | dwfRotate=12 | dwfStair=13 | dwfJTest=14 | dwfPolarity=15 | dwfMLS=16 | dwfLogSine=17}

**Example:** :DigGen:Ch(0):SignalType? 3  
[:DigGen:Ch(0):SignalType] dwfSine

**Description:** Returns the type of the Index<sup>th</sup> waveform.



## Gain?

**Command Syntax:** :DigGen:Ch(ch):Gain? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Gain] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Gain?  
[:DigGen:Ch(0):Gain] 100 pct

**Related Command(s):** Gain

**Description:** Returns the gain of the specified digital audio generator channel.

## Gain

**Command Syntax:** :DigGen:Ch(ch):Gain *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Gain 100 pct

**Related Command(s):** Gain?

**Description:** Sets the gain of the specified digital audio generator channel.

## Invert?

**Command Syntax:** :DigGen:Ch(ch):Invert?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Invert] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Invert?  
[:DigGen:Ch(0):Invert] False

**Related Command(s):** Invert

**Description:** Returns the invert/non-invert status of the digital audio generator channel.

## Invert

**Command Syntax:** :DigGen:Ch(ch):Invert *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Invert False

**Related Command(s):** Invert?

**Description:** Sets the invert/non-invert status of the digital audio generator channel.



## On?

**Command Syntax:** :DigGen:Ch(ch):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):On?  
[:DigGen:Ch(0):On] False

**Related Command(s):** On

**Description:** Returns the on/off status of the digital generator channel.

## On

**Command Syntax:** :DigGen:Ch(ch):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the digital audio generator channel..

## TotalAmpRdg?

**Command Syntax:** :DigGen:Ch(ch):TotalAmpRdg? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):TotalAmpRdg] Value

**Response Argument(s):** Value <unit>

**Example:** :DigGen:Ch(0):TotalAmpRdg?  
[:DigGen:Ch(0):TotalAmpRdg] 0.3 FFS

**Description:** Returns the total amplitude, including waveform amplitudes, EQ, channel gain, and channel on/off of the digital audio generator channel.

## 2.3.20.1.1 Sine

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Sine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator sine waveform.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):Sine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Sine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Sine(0):Amp?  
[:DigGen:Ch(0):Sine(0):Amp] 0.8 FFS

**Related Command(s):** Amp

**Description:** Queries the sine amplitude.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):Sine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Sine(0):Amp 0.8 FFS

**Related Command(s):** Amp?

**Description:** Sets the sine amplitude.

**Freq?**

**Command Syntax:** :DigGen:Ch(*ch*):Sine(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Sine(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Sine(0):Freq?  
[:DigGen:Ch(0):Sine(0):Freq] 10000.0 HZ

**Related Command(s):** Freq

**Description:** Queries the sine frequency.

## Freq

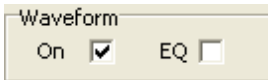
**Command Syntax:** :DigGen:Ch(ch):Sine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Sine(0):Freq 10000.0 HZ

**Related Command(s):** Freq?

**Description:** Sets the sine frequency.



## Eq?

**Command Syntax:** :DigGen:Ch(ch):Sine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Sine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Sine(0):Eq?  
 [:DigGen:Ch(0):Sine(0):Eq] False

**Related Command(s):** Eq

**Description:** Queries whether the current digital generator EQ will be applied to the sine waveform.

## Eq

**Command Syntax:** :DigGen:Ch(ch):Sine(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Sine(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets whether the current digital generator EQ will be applied to the sine waveform.

## On?

**Command Syntax:** :DigGen:Ch(ch):Sine(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Sine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Sine(0):On?  
 [:DigGen:Ch(0):Sine(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the sine waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):Sine(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Sine(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the sine waveform.

## 2.3.20.1.2 Phased Sine

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):PhasedSine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator phased sine waveform.

Sine  
 Amp: 0.0000 FFS  
 Freq: 1.00000 kHz  
 Phase: 0.0000 °

## Amp?

**Command Syntax:** :DigGen:Ch(*ch*):PhasedSine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):PhasedSine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):PhasedSine(0):Amp?  
 [:DigGen:Ch(0):PhasedSine(0):Amp] 0.5 FFS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the phased sine waveform.

## Amp

**Command Syntax:** :DigGen:Ch(*ch*):PhasedSine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):PhasedSine(0):Amp 0.5 FFS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the phased sine waveform.

## Freq?

**Command Syntax:** :DigGen:Ch(*ch*):PhasedSine(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):PhasedSine(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):PhasedSine(0):Freq?  
 [:DigGen:Ch(0):PhasedSine(0):Freq] 2000 HZ

**Related Command(s):** Freq

**Description:** Queries the frequency of the phased-sine waveform.

## Freq

**Command Syntax:** :DigGen:Ch(ch):PhasedSine(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):PhasedSine(0):Freq 2000 HZ

**Related Command(s):** Freq?

**Description:** Sets the frequency of the phased-sine waveform.

## Phase?

**Command Syntax:** :DigGen:Ch(ch):PhasedSine(i):Phase? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):PhasedSine(i):Phase] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):PhasedSine(0):Phase?  
 [:DigGen:Ch(0):PhasedSine(0):Phase] **Value**

**Related Command(s):** Phase

**Description:** Queries the phase difference between the A and B channels.

## Phase

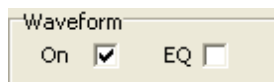
**Command Syntax:** :DigGen:Ch(ch):PhasedSine(i):Phase *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):PhasedSine(0):Phase *Value*

**Related Command(s):** Phase?

**Description:** Sets the phase difference between the A and B channels.



## Eq?

**Command Syntax:** :DigGen:Ch(ch):PhasedSine(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):PhasedSine(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):PhasedSine(0):Eq?  
 [:DigGen:Ch(0):PhasedSine(0):Eq] **False**

**Related Command(s):** Eq

**Description:** Queries whether the current generator EQ file will be applied to the phased-sine waveform.

## Eq

**Command Syntax:** `:DigGen:Ch(ch):PhasedSine(i):Eq Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):PhasedSine(0):Eq False`

**Related Command(s):** Eq?

**Description:** Sets whether the current generator EQ file will be applied to the phased-sine waveform.

## On?

**Command Syntax:** `:DigGen:Ch(ch):PhasedSine(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):PhasedSine(i):On] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):PhasedSine(0):On?`

`[ :DigGen:Ch(0):PhasedSine(0):On ] False`

**Related Command(s):** On

**Description:** Queries the on/off status of the phased-sine waveform.

## On

**Command Syntax:** `:DigGen:Ch(ch):PhasedSine(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

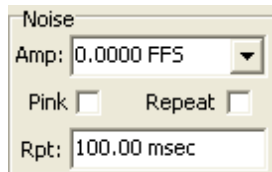
**Example:** `:DigGen:Ch(0):PhasedSine(0):On False`

**Related Command(s):** On?

**Description:** Sets the on/off status of the phased-sine waveform.

## 2.3.20.1.3 Noise

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Noise(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator noise waveform.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):Noise(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Noise(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Noise(0):Amp?

[ :DigGen:Ch(0):Noise(0):Amp] 0.5 FFS

**Related Command(s):** Amp

**Description:** Queries the peak amplitude of the noise waveform.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):Noise(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):Amp 0.5 FFS

**Related Command(s):** Amp?

**Description:** Sets the peak amplitude of the noise waveform.

**Pink?**

**Command Syntax:** :DigGen:Ch(*ch*):Noise(*i*):Pink?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(*ch*):Noise(*i*):Pink] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):Pink?

[ :DigGen:Ch(0):Noise(0):Pink] False

**Related Command(s):** Pink

**Description:** Queries whether a pink (3 dB/oct) filter is applied to the output noise waveform.



## Pink

**Command Syntax:** :DigGen:Ch(ch):Noise(i):Pink *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):Pink False

**Related Command(s):** Pink?

**Description:** Sets whether a pink (3 dB/oct) filter is applied to the output noise waveform.

## RepeatPeriod?

**Command Syntax:** :DigGen:Ch(ch):Noise(i):RepeatPeriod?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Noise(i):RepeatPeriod] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):RepeatPeriod?  
 [:DigGen:Ch(0):Noise(0):RepeatPeriod] False

**Related Command(s):** RepeatPeriod, RepeatTime

**Description:** Queries whether the noise waveform repeats after a specified period.

## RepeatPeriod

**Command Syntax:** :DigGen:Ch(ch):Noise(i):RepeatPeriod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):RepeatPeriod False

**Related Command(s):** RepeatPeriod?

**Description:** Sets whether the noise waveform repeats after a specified period.

## RepeatTime?

**Command Syntax:** :DigGen:Ch(ch):Noise(i):RepeatTime? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Noise(i):RepeatTime] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Noise(0):RepeatTime?  
 [:DigGen:Ch(0):Noise(0):RepeatTime] 0.4 s

**Related Command(s):** RepeatTime

**Description:** Queries the repeat time period used when repeat is turned on.

## RepeatTime

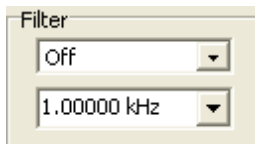
**Command Syntax:** `:DigGen:Ch(ch):Noise(i):RepeatTime Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Noise(0):RepeatTime 0.4 s`

**Related Command(s):** RepeatTime?

**Description:** Sets the repeat time period used when repeat is turned on.



## Filter?

**Command Syntax:** `:DigGen:Ch(ch):Noise(i):Filter?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):Noise(i):Filter] Value`

**Response Argument(s):** `Value <int> {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}`

**Example:** `:DigGen:Ch(0):Noise(0):Filter?`  
`[:DigGen:Ch(0):Noise(0):Filter] nsNoFilter`

**Related Command(s):** Filter

**Description:** Queries the type of filtering applied to the noise waveform.

## Filter

**Command Syntax:** `:DigGen:Ch(ch):Noise(i):Filter Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Noise(0):Filter nsNoFilter`

**Related Command(s):** Filter?

**Description:** Sets the type of filtering applied to the noise waveform.

## FilterFreq?

**Command Syntax:** `:DigGen:Ch(ch):Noise(i):FilterFreq? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:DigGen:Ch(ch):Noise(i):FilterFreq] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:DigGen:Ch(0):Noise(0):FilterFreq?`  
`[:DigGen:Ch(0):Noise(0):FilterFreq] Value`

**Related Command(s):** FilterFreq

**Description:** Queries the filter frequency.

## FilterFreq

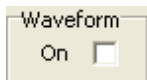
**Command Syntax:** `:DigGen:Ch(ch):Noise(i):FilterFreq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Noise(0):FilterFreq Value`

**Related Command(s):** FilterFreq?

**Description:** Sets the filter frequency.



## On?

**Command Syntax:** `:DigGen:Ch(ch):Noise(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):Noise(i):On] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Noise(0):On?`  
`[:DigGen:Ch(0):Noise(0):On] False`

**Related Command(s):** On

**Description:** Queries the on/off status of the noise waveform.

## On

**Command Syntax:** `:DigGen:Ch(ch):Noise(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

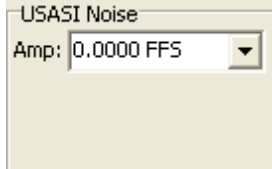
**Example:** `:DigGen:Ch(0):Noise(0):On False`

**Related Command(s):** On?

**Description:** Sets the on/off status of the noise waveform.

## 2.3.20.1.4 USASI Noise

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):USASI(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator USASI noise waveform.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):USASI(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):USASI(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):USASI(0):Amp?

[ :DigGen:Ch(0):USASI(0):Amp] 0.5 FFS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the USASI noise waveform.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):USASI(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):USASI(0):Amp 0.5 FFS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the USASI noise waveform.

**On?**

**Command Syntax:** :DigGen:Ch(*ch*):USASI(*i*):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(*ch*):USASI(*i*):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):USASI(0):On?

[ :DigGen:Ch(0):USASI(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the USASI noise waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):USASI(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):USASI(0):On False

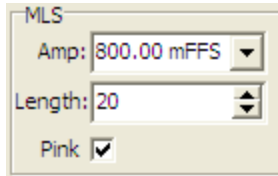
**Related Command(s):** On?

**Description:** Sets the on/off status of the USASI noise waveform.

## 2.3.20.1.5 MLS Noise

Enter topic text here.

<b>Object:</b>	<b>:DigGen:Ch(ch):MLS(i)</b>
<i>Object Argument(s):</i>	<i>ch as A or B, i as Integer</i>
<i>Description:</i>	Commands related to the digital generator MLS noise waveform.



## Amp?

**Command Syntax:** :DigGen:Ch(ch):MLS(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):MLS(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):MLS(0):Amp?  
[:DigGen:Ch(0):MLS(0):Amp] 0.8 FFS

**Related Command(s):** Amp

**Description:** Queries the sine amplitude.

## Amp

**Command Syntax:** :DigGen:Ch(ch):MLS(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):MLS(0):Amp 0.8 FFS

**Related Command(s):** Amp?

**Description:** Sets the sine amplitude.

## Length?

**Command Syntax:** :DigGen:Ch(ch):MLS(i):Length?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):MLS(i):Length] Value

**Response Argument(s):** Value <int>

**Example:** :DigGen:Ch(0):MLS(0):Length?  
[:DigGen:Ch(0):MLS(0):Length] 20

**Related Command(s):** Pink

**Description:** Queries the length of the MLS sequence.

## Length

**Command Syntax:** :DigGen:Ch(ch):Noise(i):Length Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Noise(0):Length 20

**Related Command(s):** Pink?

**Description:** Sets the length of the MLS sequence. Sequence length is  $2^{\text{length}} - 1$  samples

## Pink?

**Command Syntax:** :DigGen:Ch(ch):MLS(i):Pink?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):MLS(i):Pink] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MLS(0):Pink?  
[:DigGen:Ch(0):MLS(0):Pink] False

**Related Command(s):** Pink

**Description:** Queries whether a pink (3 dB/oct) filter is applied to the output MLS waveform.

## Pink

**Command Syntax:** :DigGen:Ch(ch):MLS(i):Pink Value [, AllowCoercion]

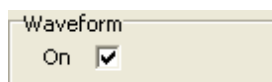
**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MLS(0):Pink False

**Related Command(s):** Pink?

**Description:** Sets whether a pink (3 dB/oct) filter is applied to the output MLS waveform.



## On?

**Command Syntax:** :DigGen:Ch(ch):MLS(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):MLS(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MLS(0):On?  
[:DigGen:Ch(0):MLS(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the MLS waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):MLS(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MLS(0):On False

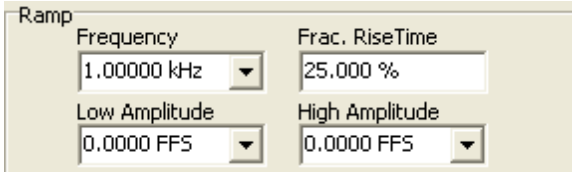
**Related Command(s):** On?

**Description:** Sets the on/off status of the MLS waveform.



2.3.20.1.6 Ramp

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Ramp(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator ramp waveform.



### Freq?

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Ramp(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Ramp(0):Freq?  
[:DigGen:Ch(0):Ramp(0):Freq] 100 HZ

**Related Command(s):** Freq

**Description:** Queries the frequency of the digital ramp waveform.

### Freq

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Ramp(0):Freq 100 HZ

**Related Command(s):** Freq?

**Description:** Sets the frequency of the digital ramp waveform.

### HiAmp?

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):HiAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Ramp(i):HiAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Ramp(0):HiAmp?  
[:DigGen:Ch(0):Ramp(0):HiAmp] 1.0 FFS

**Related Command(s):** HiAmp

**Description:** Queries the high-level amplitude of the ramp.

## HiAmp

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):HiAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Ramp(0):HiAmp 1.0 FFS

**Related Command(s):** HiAmp?

**Description:** Sets the high-level amplitude of the ramp.

## LoAmp?

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):LoAmp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Ramp(i):LoAmp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Ramp(0):LoAmp?  
 [:DigGen:Ch(0):Ramp(0):LoAmp] 0.2 FFS

**Related Command(s):** LoAmp

**Description:** Queries the low-level amplitude of the ramp.

## LoAmp

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):LoAmp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Ramp(0):LoAmp 0.2 FFS

**Related Command(s):** LoAmp?

**Description:** Sets the low-level amplitude of the ramp.

## RiseFraction?

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):RiseFraction? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Ramp(i):RiseFraction] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Ramp(0):RiseFraction?  
 [:DigGen:Ch(0):Ramp(0):RiseFraction] 25 pct

**Related Command(s):** RiseFraction

**Description:** Queries the percentage of the total waveform period that is spent going from the low to high amplitude.

## RiseFraction

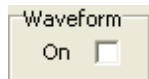
**Command Syntax:** :DigGen:Ch(ch):Ramp(i):RiseFraction *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Ramp(0):RiseFraction 25 pct

**Related Command(s):** RiseFraction?

**Description:** Sets the percentage of the total waveform period that is spent going from the low to high amplitude.



## On?

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Ramp(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Ramp(0):On?  
[:DigGen:Ch(0):Ramp(0):On] False

**Related Command(s):** On

**Description:** Queries the on/off status of the ramp waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):Ramp(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

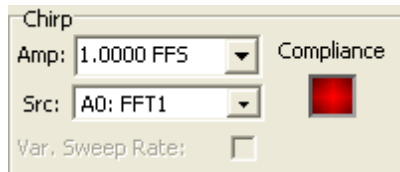
**Example:** :DigGen:Ch(0):Ramp(0):On False

**Related Command(s):** On?

**Description:** Set the on/off status of the ramp waveform.

## 2.3.20.1.7 Chirp

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Chirp(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator chirp signal.

**Amp?**

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Chirp(0):Amp?  
[:DigGen:Ch(0):Chirp(0):Amp] 0.8

**Related Command(s):** Amp

**Description:** Returns the chirp amplitude.

**Amp**

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):Amp *Value*

**Related Command(s):** Amp?

**Description:** Sets the chirp amplitude.

**FFTLINK?**

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):FFTLINK?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):FFTLINK] *Value*

**Response Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

**Example:** :DigGen:Ch(0):Chirp(0):FFTLINK?  
[:DigGen:Ch(0):Chirp(0):FFTLINK] csA0FFT

**Related Command(s):** FFTLINK

**Description:** Returns the linked FFT analyzer.

## FFTLink

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):FFTLink *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):FFTLink csA0FFT

**Related Command(s):** FFTLink?

**Description:** Sets the linked FFT analyzer

## Compliance?

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :DigGen:Ch(0):Chirp(0):Compliance?

**[:DigGen:Ch(0):Chirp(0):Compliance] scCannotGenerate**

**Description:** Returns the compliance status of the chirp waveform.

## FsMismatchRdg?

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):FsMismatchRdg?

**Command Argument(s):**

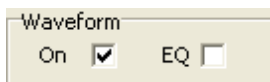
**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):FsMismatchRdg] *Value*

**Response Argument(s):** *Value* <int> {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

**Example:** :DigGen:Ch(0):Chirp(0):FsMismatchRdg?

**[:DigGen:Ch(0):Chirp(0):FsMismatchRdg] scmGenMismatched**

**Description:** Queries the ratio of the generator sampling rate to the input sampling rate of the linked analyzer. If the two are not integrally related scmGenMismatched is returned.



## On?

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):On?

**[:DigGen:Ch(0):Chirp(0):On] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the chirp waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the chirp waveform.

## Eq?

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):Eq] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):Eq?  
 [:DigGen:Ch(0):Chirp(0):Eq] False

**Related Command(s):** Eq

**Description:** Queries whether the current digital generator EQ file will be applied to the chirp waveform.

## Eq

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):Eq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets whether the current digital generator EQ file will be applied to the chirp waveform.

## VarSweep?

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):VarSweep?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Chirp(i):VarSweep] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Chirp(0):VarSweep?  
 [:DigGen:Ch(0):Chirp(0):VarSweep] False

**Related Command(s):** Eq

**Description:** Queries if variable speed sweep is enabled.

## VarSweep

**Command Syntax:** :DigGen:Ch(ch):Chirp(i):VarSweep *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

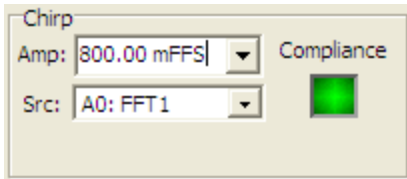
**Example:** :DigGen:Ch(0):Chirp(0):VarSweep False

**Related Command(s):** Eq?

**Description:** Sets if variable speed sweep is enabled. If enabled (true), the crest factor of a chirp with EQ is improved.

## 2.3.20.1.8 Log-Sine Chirp

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):LogSine(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator log-sine chirp signal.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):LogSine(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):LogSine(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):LogSine(0):Amp?  
[:DigGen:Ch(0):LogSine(0):Amp] 0.8

**Related Command(s):** Amp

**Description:** Returns the log-sine chirp amplitude.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):LogSine(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):LogSine(0):Amp 0.8

**Related Command(s):** Amp?

**Description:** Sets the log-sine chirp amplitude.

**FFTLINK?**

**Command Syntax:** :DigGen:Ch(*ch*):LogSine(*i*):FFTLINK?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(*ch*):LogSine(*i*):FFTLINK] *Value*

**Response Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

**Example:** :DigGen:Ch(0):LogSine(0):FFTLINK?  
[:DigGen:Ch(0):LogSine(0):FFTLINK] csA0FFT

**Related Command(s):** FFTLINK

**Description:** Returns the linked FFT analyzer.



## FFTLink

**Command Syntax:** :DigGen:Ch(ch):LogSine(i):FFTLink *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):LogSine(0):FFTLink csA0FFT

**Related Command(s):** FFFLink?

**Description:** Sets the linked FFT analyzer

## Compliance?

**Command Syntax:** :DigGen:Ch(ch):LogSine(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):LogSine(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :DigGen:Ch(0):LogSine(0):Compliance?

**[:DigGen:Ch(0):LogSine(0):Compliance] scCannotGenerate**

**Description:** Returns the compliance status of the log-sine chirp waveform.

## FsMismatchRdg?

**Command Syntax:** :DigGen:Ch(ch):LogSine(i):FsMismatchRdg?

**Command Argument(s):**

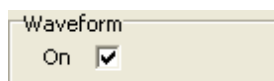
**Response Syntax:** [:DigGen:Ch(ch):LogSine(i):FsMismatchRdg] *Value*

**Response Argument(s):** *Value* <int> {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

**Example:** :DigGen:Ch(0):LogSine(0):FsMismatchRdg?

**[:DigGen:Ch(0):LogSine(0):FsMismatchRdg] scmGenMismatched**

**Description:** Queries the ratio of the generator sampling rate to the input sampling rate of the linked analyzer. If the two are not integrally related scmGenMismatched is returned.



## On?

**Command Syntax:** :DigGen:Ch(ch):LogSine(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):LogSine(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):LogSine(0):On?

**[:DigGen:Ch(0):LogSine(0):On] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the log-sine chirp waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):LogSine(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

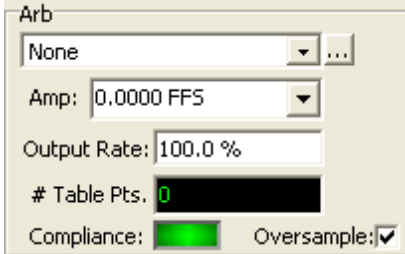
**Example:** :DigGen:Ch(0):LogSine(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the log-sine chirp waveform.

2.3.20.1.9 Arbitrary

<b>Object:</b>	<b>:DigGen:Ch(ch):Arb(i)</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the digital generator arbitrary waveform.



### Amp?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Arb(0):Amp?  
[:DigGen:Ch(0):Arb(0):Amp] 0.8 FFS

**Related Command(s):** Amp

**Description:** Returns the amplitude of the arbitrary waveform.

### Amp

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Arb(0):Amp 0.8 FFS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the arbitrary waveform.

### Compliance?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Compliance?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :DigGen:Ch(0):Arb(0):Compliance?  
[:DigGen:Ch(0):Arb(0):Compliance] scCannotGenerate

**Description:** Returns the compliance status of the arbitrary waveform.

## NumPointsRdg?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):NumPointsRdg?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):NumPointsRdg] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigGen:Ch(0):Arb(0):NumPointsRdg?  
[:DigGen:Ch(0):Arb(0):NumPointsRdg] 1024

**Description:** Queries the number of points the arbitrary waveform file.

## Oversample?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Oversample?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):Oversample] *Value*

**Response Argument(s):** *Value* <int> {False=0|True=1}

**Example:** :DigGen:Ch(0):Arb(0):Oversample?  
[:DigGen:Ch(0):Arb(0):Oversample] False

**Related Command(s):** Oversample

**Description:** Queries whether the arbitrary waveform file points are oversampled before output.

## Oversample

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Oversample *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0|True=1}

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Arb(0):Oversample False

**Related Command(s):** Oversample?

**Description:** Sets whether the arbitrary waveform file points are oversampled before output.

## RateMultiplier?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):RateMultiplier? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):RateMultiplier] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Arb(0):RateMultiplier?  
[:DigGen:Ch(0):Arb(0):RateMultiplier] 100 pct

**Related Command(s):** RateMultiplier

**Description:** Queries the output rate (in table points per sample).

## RateMultiplier

**Command Syntax:** :DigGen:Ch(ch):Arb(i):RateMultiplier *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Arb(0):RateMultiplier 50 pct

**Related Command(s):** RateMultiplier?

**Description:** Sets the output rate (in table points per sample).

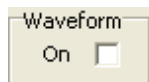
## Load

**Command Syntax:** :DigGen:Ch(ch):Arb(i):Load *FileName*, *ColSelect*

**Command Argument(s):** *FileName* <string>  
*ColSelect* <int>

**Example:** :DigGen:Ch(0):Arb(0):Load "MyFile.arb", 0

**Description:** Loads the specified column from the specified arbitrary waveform file.



## On?

**Command Syntax:** :DigGen:Ch(ch):Arb(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Arb(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Arb(0):On?  
 [:DigGen:Ch(0):Arb(0):On] False

**Related Command(s):** On

**Description:** Returns the on/off status of the Arb waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):Arb(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

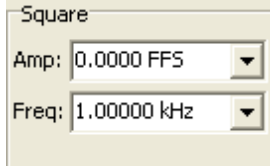
**Example:** :DigGen:Ch(0):Arb(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the Arb waveform.

## 2.3.20.1.10 Square

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Square(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator square-wave.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):Square(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Square(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Square(0):Amp?  
[:DigGen:Ch(0):Square(0):Amp] 0.8

**Related Command(s):** Amp

**Description:** Queries the square-wave amplitude.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):Square(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):Amp 0.8

**Related Command(s):** Amp?

**Description:** Sets the square-wave amplitude.

**Freq?**

**Command Syntax:** :DigGen:Ch(*ch*):Square(*i*):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Square(*i*):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Square(0):Freq?  
[:DigGen:Ch(0):Square(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Queries the square-wave frequency.

## Freq

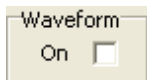
**Command Syntax:** :DigGen:Ch(ch):Square(i):Freq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):Freq 1000 HZ

**Related Command(s):** Freq?

**Description:** Sets the square-wave frequency.



## On?

**Command Syntax:** :DigGen:Ch(ch):Square(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Square(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):On?  
[:DigGen:Ch(0):Square(0):On] **False**

**Related Command(s):** On

**Description:** Returns the on/off status of the square-wave waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):Square(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the square-wave waveform.

## PrecisionSquare?

**Command Syntax:** :DigGen:Ch(ch):Square(i):PrecisionSquare?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Square(i):PrecisionSquare] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):PrecisionSquare?  
[:DigGen:Ch(0):Square(0):PrecisionSquare] **False**

**Related Command(s):** PrecisionSquare

**Description:** Queries whether the square-wave frequency will be limited to "precision frequencies", i.e. frequencies where there are equal integer numbers of up and down samples.

## PrecisionSquare

**Command Syntax:** :DigGen:Ch(ch):Square(i):PrecisionSquare *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Square(0):PrecisionSquare False

**Related Command(s):** PrecisionSquare?

**Description:** Sets whether the square-wave frequency will be limited to "precision frequencies", i.e. frequencies where there are equal integer numbers of up and down samples.



2.3.20.1.11 IMD

<b>Object:</b>	<b>:DigGen:Ch(ch):IMDSig(i)</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the digital IMD signal.

IMD

Total Amp 1.0000 FFS

High Freq. 14.2857 kHz

IM Freq. 1.00000 kHz

Amp. Ratio 4:1

IMD

Total Amp 1.0000 FFS

Center Freq. 14.2857 kHz

IM Freq. 1.00000 kHz

Amp. Ratio 1:1

IMD

Total Amp 1.0000 FFS

Sine Freq. 14.2857 kHz

Sqr. Freq. 3.00000 kHz

Amp. Ratio 4:1

## Amp?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):IMDSig(0):Amp?  
[:DigGen:Ch(0):IMDSig(0):Amp] 0.8 FFS

**Related Command(s):** Amp

**Description:** Queries the total amplitude of the IMD waveform.

## Amp

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):Amp 0.8 FFS

**Related Command(s):** Amp?

**Description:** Sets the total amplitude of the IMD waveform.

## AmpRatio?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):AmpRatio?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):AmpRatio] Value

**Response Argument(s):** Value <int> {ar41=0 | ar11=1}

**Example:** :DigGen:Ch(0):IMDSig(0):AmpRatio?  
[:DigGen:Ch(0):IMDSig(0):AmpRatio] ar41

**Related Command(s):** AmpRatio

**Description:** Queries the tone amplitude ratio for IMD waveforms.

## AmpRatio

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):AmpRatio Value [, AllowCoercion]

**Command Argument(s):** Value <int> {ar41=0 | ar11=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):AmpRatio ar41

**Related Command(s):** AmpRatio?

**Description:** Sets the tone amplitude ratio for the IMD waveform. The amplitude ratio is only variable in SMPTE mode.

## IMFreq?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):IMFreq? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):IMFreq] Value

**Response Argument(s):** Value <unit>

**Example:** :DigGen:Ch(0):IMDSig(0):IMFreq?  
[:DigGen:Ch(0):IMDSig(0):IMFreq] 1000 HZ

**Related Command(s):** IMFreq

**Description:** Queries the "IMD" frequency of the IMD waveform. The "IM" frequency is the Low Frequency in SMPTE mode, the difference frequency in CCIF mode, and the Square Wave Frequency in DIM mode.

## IMFreq

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):IMFreq Value [, AllowCoercion]

**Command Argument(s):** Value <unit>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):IMFreq 1000 HZ

**Related Command(s):** IMFreq?

**Description:** Sets the "IMD" frequency of the IMD waveform. The "IM" frequency is the Low Frequency in SMPTE mode, the difference frequency in CCIF mode, and the Square Wave Frequency in DIM mode.

## MainFreq?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):MainFreq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):MainFreq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):IMDSig(0):MainFreq?  
[:DigGen:Ch(0):IMDSig(0):MainFreq] 14000 HZ

**Related Command(s):** MainFreq

**Description:** Queries the "Main" frequency of the IMD waveform. The "Main" frequency is the High Frequency in SMPTE mode, the center frequency in CCIF mode, and the Sine Frequency in DIM mode.

## MainFreq

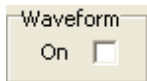
**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):MainFreq *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):MainFreq 14000 HZ

**Related Command(s):** MainFreq?

**Description:** Sets the "Main" frequency of the IMD waveform. The "Main" frequency is the High Frequency in SMPTE mode, the center frequency in CCIF mode, and the Sine Frequency in DIM mode.



## On?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):On?  
[:DigGen:Ch(0):IMDSig(0):On] False

**Related Command(s):** On

**Description:** Returns the on/off status of the IMD waveform.

## On

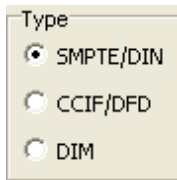
**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):IMDSig(0):On False

**Related Command(s):** On?

**Description:** Sets the the on/off status of the IMD waveform.



## Type?

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):Type?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):IMDSig(i):Type] Value

**Response Argument(s):** Value <int> {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

**Example:** :DigGen:Ch(0):IMDSig(0):Type?  
[:DigGen:Ch(0):IMDSig(0):Type] imdSMPTE

**Related Command(s):** Type

**Description:** Returns the type of IMD waveform.

## Type

**Command Syntax:** :DigGen:Ch(ch):IMDSig(i):Type Value [, AllowCoercion]

**Command Argument(s):** Value <int> {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

AllowCoercion <bool> {False=0 | True=1}

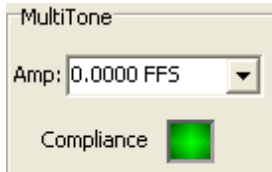
**Example:** :DigGen:Ch(0):IMDSig(0):Type imdSMPTE

**Related Command(s):** Type?

**Description:** Sets the type of the IMD waveform.

2.3.20.1.12 MultiTone

<b>Object:</b>	<b>:DigGen:Ch(ch):MultiTone(i)</b>
<i>Object Argument(s):</i>	<i>i &lt;int&gt;</i>
<i>Description:</i>	Commands related to the digital generator multitone signal.



### Amp?

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):MultiTone(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):MultiTone(0):Amp?

[ :DigGen:Ch(0):MultiTone(0):Amp] 0.8 FFS

**Related Command(s):** Amp

**Description:** .Queries the multitone waveform amplitude.

### Amp

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MultiTone(0):Amp 0.8 FFS

**Related Command(s):** Amp?

**Description:** Sets the multitone waveeform amplitude.

### Compliance?

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):Compliance?

**Command Argument(s):**

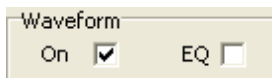
**Response Syntax:** [:DigGen:Ch(ch):MultiTone(i):Compliance] *Value*

**Response Argument(s):** *Value* <int> {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

**Example:** :DigGen:Ch(0):MultiTone(0):Compliance?

[ :DigGen:Ch(0):MultiTone(0):Compliance] scCannotGenerate

**Description:** Queries the compliance status of the multitone waveform.



## Eq?

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):Eq?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):MultiTone(i):Eq] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MultiTone(0):Eq?  
[:DigGen:Ch(0):MultiTone(0):Eq] False

**Related Command(s):** Eq

**Description:** Returns whether the currently selected EQ file will be applied to the multitone signal.

## Eq

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):Eq Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MultiTone(0):Eq False

**Related Command(s):** Eq?

**Description:** Sets whether the currently selected EQ file will be applied to the multitone signal..

## On?

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):MultiTone(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MultiTone(0):On?  
[:DigGen:Ch(0):MultiTone(0):On] False

**Related Command(s):** On

**Description:** Returns the on/off status of the multitone waveform.

## On

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):MultiTone(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the multitone waveform.

Export

## ExportMultiTone

**Command Syntax:** :DigGen:Ch(ch):MultiTone(i):ExportMultiTone *FileName, Mode, NumBits, Dither*

**Command Argument(s):** *FileName* <string>

*Mode* <int> {mtArbFile=0 | mtWavFile=1}

*NumBits* <int>

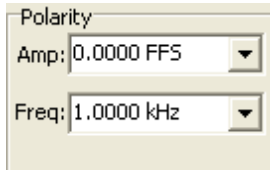
*Dither* <int> {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

**Example:** :DigGen:Ch(0):MultiTone(0):ExportMultiTone Value, mtArbF

**Description:** Exports the current multitone signal to a file with the specified filename. The mode argument specifies whether the file will be a .WAV file or an SRI arbitrary waveform file. The Dither argument specifies whether the output file will include dither.

## 2.3.20.1.13 Polarity

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Polarity(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator Polarity signal.

**Amp?**

**Command Syntax:** :DigGen:Ch(ch):Polarity(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Polarity(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Polarity(0):Amp?

[ :DigGen:Ch(0):Polarity(0):Amp] .8 FFS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the polarity waveform.

**Amp**

**Command Syntax:** :DigGen:Ch(ch):Polarity(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Polarity(0):Amp .8 FFS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the polarity waveform.

**Freq?**

**Command Syntax:** :DigGen:Ch(ch):Polarity(i):Freq? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Polarity(i):Freq] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Polarity(0):Freq?

[ :DigGen:Ch(0):Polarity(0):Freq] 1000 HZ

**Related Command(s):** Freq

**Description:** Returns the frequency of the polarity waveform.



## Freq

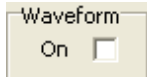
**Command Syntax:** `:DigGen:Ch(ch):Polarity(i):Freq Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Polarity(0):Freq 1000 HZ`

**Related Command(s):** Freq?

**Description:** Sets the frequency of the polarity waveform.



## On?

**Command Syntax:** `:DigGen:Ch(ch):Polarity(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):Polarity(i):On] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Polarity(0):On?`  
`[:DigGen:Ch(0):Polarity(0):On] False`

**Related Command(s):** On

**Description:** Returns the on/off status of the polarity signal.

## On

**Command Syntax:** `:DigGen:Ch(ch):Polarity(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Polarity(0):On False`

**Related Command(s):** On?

**Description:** Sets the on/off status of the polarity waveform.

## 2.3.20.1.14 Count

<b>Object:</b>	<b>:DigGen:Ch(ch):Count(i)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator counter waveform.

## Amp?

**Command Syntax:** :DigGen:Ch(ch):Count(i):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Count(i):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Count(0):Amp?  
[:DigGen:Ch(0):Count(0):Amp] 0.0 FFS

**Related Command(s):** Amp

**Description:** Queries the initial value of the digital counter signal.

## Amp

**Command Syntax:** :DigGen:Ch(ch):Count(i):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):Amp 0.0 FFS

**Related Command(s):** Amp?

**Description:** Sets the initial value of the digital counter signal.

## Polarity1?

**Command Syntax:** :DigGen:Ch(ch):Count(i):Polarity1?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Count(i):Polarity1] *Value*

**Response Argument(s):** *Value* <int> {Positive=0|Negative=1}

**Example:** :DigGen:Ch(0):Count(0):Polarity1?  
[:DigGen:Ch(0):Count(0):Polarity1] Positive

**Related Command(s):** Polarity1

**Description:** Queries the polarity of the initial count value.

## Polarity1

**Command Syntax:** :DigGen:Ch(ch):Count(i):Polarity1 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {Positive=0|Negative=1}  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):Polarity1 Positive

**Related Command(s):** Polarity1?

**Description:** Sets the polarity of the initial count value.

## Amp2?

**Command Syntax:** :DigGen:Ch(ch):Count(i):Amp2? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(ch):Count(i):Amp2] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Count(0):Amp2?  
 [:DigGen:Ch(0):Count(0):Amp2] 0.5 FFS

**Related Command(s):** Amp2

**Description:** Queries the final value of the digital counter signal.

## Amp2

**Command Syntax:** :DigGen:Ch(ch):Count(i):Amp2 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):Amp2 0.5 FFS

**Related Command(s):** Amp2?

**Description:** Sets the final value of the digital counter signal.

## Polarity2?

**Command Syntax:** :DigGen:Ch(ch):Count(i):Polarity2?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Count(i):Polarity2] *Value*

**Response Argument(s):** *Value* <int> {Positive=0|Negative=1}

**Example:** :DigGen:Ch(0):Count(0):Polarity2?  
 [:DigGen:Ch(0):Count(0):Polarity2] Positive

**Related Command(s):** Polarity2

**Description:** Queries the polarity of the final count value.

## Polarity2

**Command Syntax:** :DigGen:Ch(ch):Count(i):Polarity2 *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {Positive=0|Negative=1}  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):Polarity2 Positive

**Related Command(s):** Polarity2?

**Description:** Sets the polarity of the final count value.

## Dwell?

**Command Syntax:** :DigGen:Ch(ch):Count(i):Dwell?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Count(i):Dwell] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigGen:Ch(0):Count(0):Dwell?  
 [:DigGen:Ch(0):Count(0):Dwell] **Value**

**Related Command(s):** Dwell

**Description:** Queries the number of samples the signal outputs each count value.

## Dwell

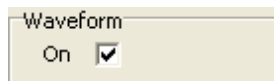
**Command Syntax:** :DigGen:Ch(ch):Count(i):Dwell *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):Dwell *Value*

**Related Command(s):** Dwell?

**Description:** Sets the number of samples the signal outputs each count value.



## On?

**Command Syntax:** :DigGen:Ch(ch):Count(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Count(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0|True=1}

**Example:** :DigGen:Ch(0):Count(0):On?  
 [:DigGen:Ch(0):Count(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the digital count signal.

## On

**Command Syntax:** :DigGen:Ch(ch):Count(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

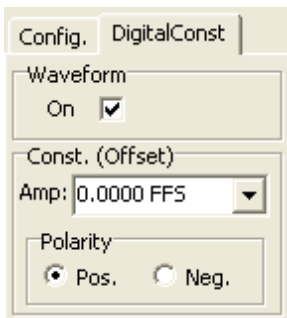
**Example:** :DigGen:Ch(0):Count(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the digital count signal.

## 2.3.20.1.15 Digital Constant

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Constant(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator constant signal.

**Amp?**

**Command Syntax:** :DigGen:Ch(*ch*):Constant(*i*):Amp? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:DigGen:Ch(*ch*):Constant(*i*):Amp] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :DigGen:Ch(0):Constant(0):Amp?

[:DigGen:Ch(0):Constant(0):Amp] 0.334 FFS

**Related Command(s):** Amp

**Description:** Queries the amplitude of the digital constant.

**Amp**

**Command Syntax:** :DigGen:Ch(*ch*):Constant(*i*):Amp *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :DigGen:Ch(0):Constant(0):Amp 0.334 FFS

**Related Command(s):** Amp?

**Description:** Sets the amplitude of the digital constant. Note that the amplitude is always a positive value—the sign of the output is set with the :Polarity command.

## On?

**Command Syntax:** `:DigGen:Ch(ch):Constant(i):On?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):Constant(i):On] Value`

**Response Argument(s):** `Value <int> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Constant(0):On?`

`[:DigGen:Ch(0):Constant(0):On] False`

**Related Command(s):** On

**Description:** Queries the on/off status of the digital constant signal.

## On

**Command Syntax:** `:DigGen:Ch(ch):Constant(i):On Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {False=0 | True=1}`

`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:DigGen:Ch(0):Constant(0):On False`

**Related Command(s):** On?

**Description:** Sets the on/off status of the digital constant signal.

## Polarity?

**Command Syntax:** `:DigGen:Ch(ch):Constant(i):Polarity?`

**Command Argument(s):**

**Response Syntax:** `[:DigGen:Ch(ch):Constant(i):Polarity] Value`

**Response Argument(s):** `Value <int> {Positive=0 | Negative=1}`

**Example:** `:DigGen:Ch(0):Constant(0):Polarity?`

`[:DigGen:Ch(0):Constant(0):Polarity] Positive`

**Related Command(s):** Polarity

**Description:** Queries the polarity of the digital constant signal.

## Polarity

**Command Syntax:** `:DigGen:Ch(ch):Constant(i):Polarity Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {Positive=0 | Negative=1}`

`AllowCoercion <bool> {False=0 | True=1}`

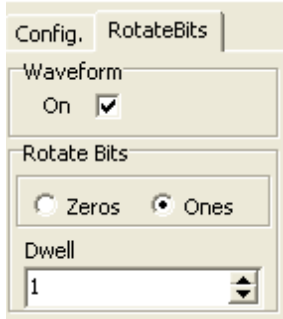
**Example:** `:DigGen:Ch(0):Constant(0):Polarity Positive`

**Related Command(s):** Polarity?

**Description:** Sets the polarity of the digital constant signal.

## 2.3.20.1.16 Rotate

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Rotate(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator rotate bits signal.

**Dwell?**

**Command Syntax:** :DigGen:Ch(*ch*):Rotate(*i*):Dwell?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(*ch*):Rotate(*i*):Dwell] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigGen:Ch(0):Rotate(0):Dwell?

[ :DigGen:Ch(0):Rotate(0):Dwell] **Value**

**Related Command(s):** Dwell

**Description:** Queries the number of samples the signal outputs each bit pattern for.

**Dwell**

**Command Syntax:** :DigGen:Ch(*ch*):Rotate(*i*):Dwell *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Rotate(0):Dwell *Value*

**Related Command(s):** Dwell?

**Description:** Sets the number of samples the signal outputs each bit pattern for.



## On?

**Command Syntax:** :DigGen:Ch(ch):Rotate(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Rotate(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Rotate(0):On?  
[:DigGen:Ch(0):Rotate(0):On] **False**

**Related Command(s):** On

**Description:** Queries the on/off status of the signal.

## On

**Command Syntax:** :DigGen:Ch(ch):Rotate(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Rotate(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the signal.

## WalkVal?

**Command Syntax:** :DigGen:Ch(ch):Rotate(i):WalkVal?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Rotate(i):WalkVal] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigGen:Ch(0):Rotate(0):WalkVal?  
[:DigGen:Ch(0):Rotate(0):WalkVal] **Value**

**Related Command(s):** WalkVal

**Description:** Queries whether the signal is walking ones (returns 1) or walking zeros (returns 0).

## WalkVal

**Command Syntax:** :DigGen:Ch(ch):Rotate(i):WalkVal *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

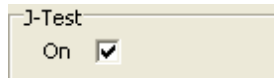
**Example:** :DigGen:Ch(0):Rotate(0):WalkVal **Value**

**Related Command(s):** WalkVal?

**Description:** Sets whether the signal is walking ones (Value = 1) or walking zeros (Value = 0).

## 2.3.20.1.17 JTest

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):JTest(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>ch</i> <char> {A   B}, <i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator J-Test signal.

**On?**

**Command Syntax:** :DigGen:Ch(ch):JTest(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):JTest(i):On] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):JTest(0):On?  
[:DigGen:Ch(0):JTest(0):On] **False**

**Related Command(s):** On

**Description:** Returns the on/off status of the J-Test signal.

**On**

**Command Syntax:** :DigGen:Ch(ch):JTest(i):On *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

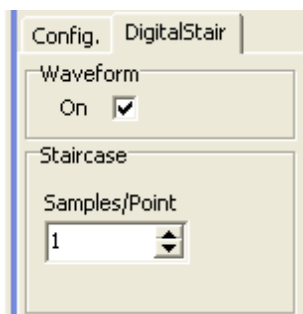
**Example:** :DigGen:Ch(0):JTest(0):On **False**

**Related Command(s):** On?

**Description:** Sets the on/off status of the J-Test signal.

2.3.20.1.18 Stairstep

<b>Object:</b>	<b>:DigGen:Ch(<i>ch</i>):Stair(<i>i</i>)</b>
<i>Object Argument(s):</i>	<i>i</i> <int>
<i>Description:</i>	Commands related to the digital generator stairstep signal.



## Dwell?

**Command Syntax:** :DigGen:Ch(*ch*):Stair(*i*):Dwell?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(*ch*):Stair(*i*):Dwell] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :DigGen:Ch(0):Stair(0):Dwell?  
[:DigGen:Ch(0):Stair(0):Dwell] 20

**Related Command(s):** Dwell

**Description:** Queries the number of samples the stairstep signal outputs at each step value.

## Dwell

**Command Syntax:** :DigGen:Ch(*ch*):Stair(*i*):Dwell *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Stair(0):Dwell 20

**Related Command(s):** Dwell?

**Description:** Sets the number of samples the stairstep signal outputs at each step value.

## On?

**Command Syntax:** :DigGen:Ch(ch):Stair(i):On?

**Command Argument(s):**

**Response Syntax:** [:DigGen:Ch(ch):Stair(i):On] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Stair(0):On?

**[ :DigGen:Ch(0):Stair(0):On ] False**

**Related Command(s):** On

**Description:** Queries the on/off status of the stairstep signal.

## On

**Command Syntax:** :DigGen:Ch(ch):Stair(i):On Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :DigGen:Ch(0):Stair(0):On False

**Related Command(s):** On?

**Description:** Sets the on/off status of the stairstep signal.

### 2.3.21 Quick Measurements

<b>Object:</b>	<b>:QuickMeas</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the Quick Measurements.

Output

Domain: Digital    Connector: XLR

Channels: 2    Fs: 48.000 kHz

Dual Conn.

## OutDomain?

**Command Syntax:** :QuickMeas:OutDomain?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:OutDomain] Value

**Response Argument(s):** Value <int> {odAnalog=0 | odDigital=1 | odExternal=2}

**Example:** :QuickMeas:OutDomain?

**[:QuickMeas:OutDomain] odAnalog**

**Related Command(s):** OutDomain

**Description:** Queries the output domain, analog or digital, for quick measurements.

## OutDomain

**Command Syntax:** :QuickMeas:OutDomain Value [, AllowCoercion]

**Command Argument(s):** Value <int> {odAnalog=0 | odDigital=1 | odExternal=2}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutDomain odAnalog

**Related Command(s):** OutDomain?

**Description:** Sets the output domain, analog or digital, for quick measurements.

## OutChannels?

**Command Syntax:** :QuickMeas:OutChannels?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:OutChannels] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:OutChannels?  
[:QuickMeas:OutChannels] 1

**Related Command(s):** OutChannels

**Description:** Queries the number of output channels for quick measurements.

## OutChannels

**Command Syntax:** :QuickMeas:OutChannels *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutChannels *Value*

**Related Command(s):** OutChannels?

**Description:** Sets the number of output channels for quick measurements.

## OutConnector?

**Command Syntax:** :QuickMeas:OutConnector?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:OutConnector] *Value*

**Response Argument(s):** *Value* <int> {cXLR=0 | cBNC=1 | cOptical=2}

**Example:** :QuickMeas:OutConnector?  
[:QuickMeas:OutConnector] cXLR

**Related Command(s):** OutConnector

**Description:** Queries the output connector type for quick measurements.

## OutConnector

**Command Syntax:** :QuickMeas:OutConnector *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cXLR=0 | cBNC=1 | cOptical=2}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutConnector cXLR

**Related Command(s):** OutConnector?

**Description:** Sets the output connector type for quick measurements.

## OutAnlgBW?

**Command Syntax:** :QuickMeas:OutAnlgBW?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:OutAnlgBW] Value

**Response Argument(s):** Value <int> {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}

**Example:** :QuickMeas:OutAnlgBW?

**[ :QuickMeas:OutAnlgBW ] bw200kHz**

**Related Command(s):** OutAnlgBW

**Description:** Queries the output bandwidth selection for analog output quick measurements.

## OutAnlgBW

**Command Syntax:** :QuickMeas:OutAnlgBW Value [, AllowCoercion]

**Command Argument(s):** Value <int> {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutAnlgBW bw200kHz

**Related Command(s):** OutAnlgBW?

**Description:** Sets the output bandwidth selection for analog output quick measurements.

## OutDigDualConn?

**Command Syntax:** :QuickMeas:OutDigDualConn?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:OutDigDualConn] Value

**Response Argument(s):** Value <int> {False=0 | True=1}

**Example:** :QuickMeas:OutDigDualConn?

**[ :QuickMeas:OutDigDualConn ] False**

**Related Command(s):** OutDigDualConn

**Description:** Queries whether dual-connector mode is used when the output domain is digital.

## OutDigDualConn

**Command Syntax:** :QuickMeas:OutDigDualConn Value [, AllowCoercion]

**Command Argument(s):** Value <int> {False=0 | True=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutDigDualConn False

**Related Command(s):** OutDigDualConn?

**Description:** Sets whether dual-connector mode is used when the output domain is digital.

## OutDigFs?

**Command Syntax:** :QuickMeas:OutDigFs? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:OutDigFs] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:OutDigFs?  
[:QuickMeas:OutDigFs] **Value**

**Related Command(s):** OutDigFs

**Description:** Queries the output sample rate for digital-domain output quick measurements.

## OutDigFs

**Command Syntax:** :QuickMeas:OutDigFs *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:OutDigFs *Value*

**Related Command(s):** OutDigFs?

**Description:** Sets the output sample rate for digital-domain output quick measurements.

## InDomain?

**Command Syntax:** :QuickMeas:InDomain?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InDomain] *Value*

**Response Argument(s):** *Value* <int> {idAnalog=0 | idDigital=1 | idGenMon=2}

**Example:** :QuickMeas:InDomain?  
[:QuickMeas:InDomain] **idAnalog**

**Related Command(s):** InDomain

**Description:** Queries the input domain (analog or digital) for quick measurements.



## InDomain

**Command Syntax:** :QuickMeas:InDomain *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {idAnalog=0 | idDigital=1 | idGenMon=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InDomain idAnalog

**Related Command(s):** InDomain?

**Description:** Sets the input domain (analog or digital) for quick measurements.

## InChannels?

**Command Syntax:** :QuickMeas:InChannels?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InChannels] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InChannels?  
 [:QuickMeas:InChannels] **Value**

**Related Command(s):** InChannels

**Description:** Queries the number of input channels.

## InChannels

**Command Syntax:** :QuickMeas:InChannels *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InChannels *Value*

**Related Command(s):** InChannels?

**Description:** Sets the number of input channels.

## InConnector?

**Command Syntax:** :QuickMeas:InConnector?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InConnector] *Value*

**Response Argument(s):** *Value* <int> {cXLR=0 | cBNC=1 | cOptical=2}

**Example:** :QuickMeas:InConnector?  
 [:QuickMeas:InConnector] **cXLR**

**Related Command(s):** InConnector

**Description:** Queries the input connector type.

## InConnector

**Command Syntax:** :QuickMeas:InConnector *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cXLR=0 | cBNC=1 | cOptical=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InConnector cXLR

**Related Command(s):** InConnector?

**Description:** Sets the input connector type.

## InCoupling?

**Command Syntax:** :QuickMeas:InCoupling?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InCoupling] *Value*

**Response Argument(s):** *Value* <int> {cplAC=0 | cplDC=1}

**Example:** :QuickMeas:InCoupling?  
 [:QuickMeas:InCoupling] cplAC

**Related Command(s):** InCoupling

**Description:** Queries the input coupling mode.

## InCoupling

**Command Syntax:** :QuickMeas:InCoupling *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {cplAC=0 | cplDC=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InCoupling cplAC

**Related Command(s):** InCoupling?

**Description:** Sets the input coupling mode.

## InAnlgBW?

**Command Syntax:** :QuickMeas:InAnlgBW?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InAnlgBW] *Value*

**Response Argument(s):** *Value* <int> {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}

**Example:** :QuickMeas:InAnlgBW?  
 [:QuickMeas:InAnlgBW] bw200kHz

**Related Command(s):** InAnlgBW

**Description:** Queries the analog bandwidth selection for analog-input quick measurements.

## InAnlgBW

**Command Syntax:** :QuickMeas:InAnlgBW *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InAnlgBW bw200kHz

**Related Command(s):** InAnlgBW?

**Description:** Sets the analog bandwidth selection for analog-input quick measurements.

## InAutoscale?

**Command Syntax:** :QuickMeas:InAutoscale?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InAutoscale] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :QuickMeas:InAutoscale?  
 [:QuickMeas:InAutoscale] False

**Related Command(s):** InAutoscale

**Description:** Queries whether input autoranging is on or off during quick measurements.

## InAutoscale

**Command Syntax:** :QuickMeas:InAutoscale *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InAutoscale False

**Related Command(s):** InAutoscale?

**Description:** Sets whether input autoranging is on or off during quick measurements.

## InDigDualConn?

**Command Syntax:** :QuickMeas:InDigDualConn?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InDigDualConn] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :QuickMeas:InDigDualConn?  
 [:QuickMeas:InDigDualConn] False

**Related Command(s):** InDigDualConn

**Description:** Queries whether dual-connector mode is used for digital input quick measurements.

## InDigDualConn

**Command Syntax:** :QuickMeas:InDigDualConn *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InDigDualConn False

**Related Command(s):** InDigDualConn?

**Description:** Sets whether dual-connector mode is used for digital input quick measurements.

## InTermination?

**Command Syntax:** :QuickMeas:InTermination?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InTermination] *Value*

**Response Argument(s):** *Value* <int> {False=0 | True=1}

**Example:** :QuickMeas:InTermination?  
 [:QuickMeas:InTermination] False

**Related Command(s):** InTermination

**Description:** Queries whether input termination is applied for digital input quick measurements.

## InTermination

**Command Syntax:** :QuickMeas:InTermination *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {False=0 | True=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InTermination False

**Related Command(s):** InTermination?

**Description:** Sets whether input termination is applied for digital input quick measurements.

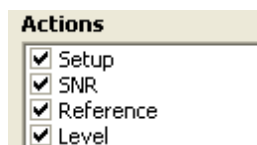
## Setup

**Command Syntax:** :QuickMeas:Setup

**Command Argument(s):** None

**Example:** :QuickMeas:Setup

**Description:** Sets up the instrument according to the values entered on the Quick Measurements Setup panel.



## AutomMeasConfig?

**Command Syntax:** :QuickMeas:AutomMeasConfig?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:AutomMeasConfig] Value

**Response Argument(s):** Value <int> {amcSetup=1 | amcSNR=2 | amcRef=4 | amcLevel=8 | amcTHDN=16 | amcFreqResp=32 | amcDistortion=64 | amcIMD=128 | amcCrosstalk=256 | amcInterChPhase=512 | amcInOutPhase=1024 | amcOpenReport=2048}

**Example:** :QuickMeas:AutomMeasConfig?  
[:QuickMeas:AutomMeasConfig] 19

**Related Command(s):** AutomMeasConfig

**Description:** Returns an integer representing the measurements included in the automated measurement set. Each measurement is assigned a bit-value as described above. The returned integer is the sum of the values corresponding to the included measurements.

## AutomMeasConfig

**Command Syntax:** :QuickMeas:AutomMeasConfig Value [, AllowCoercion]

**Command Argument(s):** Value <int> {amcSetup=1 | amcSNR=2 | amcRef=4 | amcLevel=8 | amcTHDN=16 | amcFreqResp=32 | amcDistortion=64 | amcIMD=128 | amcCrosstalk=256 | amcInterChPhase=512 | amcInOutPhase=1024 | amcOpenReport=2048}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:AutomMeasConfig 19

**Related Command(s):** AutomMeasConfig?

**Description:** Sets the measurements included in the automated measurement set. The argument is an integer representing the measurements included in the set. Each measurement is assigned a bit-value as described above. The input argument is the sum of the values corresponding to the included measurements.

## GetAutomMeasStatus?

**Command Syntax:** :QuickMeas:GetAutomMeasStatus?

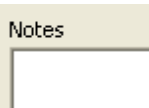
**Command Argument(s):** None

**Response Syntax:** [:QuickMeas:GetAutomMeasStatus] Status

**Response Argument(s):** Status <int> {amsIdle=0 | amsSetup=1 | amsSNR=2 | amsRef=4 | amsLevel=8 | amsTHDN=16 | amsFreqResp=32 | amsDistortion=64 | amsIMD=128 | amsCrosstalk=256 | amsInterChPhase=512}

**Example:** :QuickMeas:GetAutomMeasStatus?  
[:QuickMeas:GetAutomMeasStatus] amsIdle

**Description:** Returns a value corresponding to the automated measurement currently being executed. If no measurement is being executed the command returns amsIdle (0).



## Notes?

**Command Syntax:** :QuickMeas:Notes?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Notes] Value

**Response Argument(s):** Value <string>

**Example:** :QuickMeas:Notes?  
[:QuickMeas:Notes] Value

**Related Command(s):** Notes

**Description:** Queries the string corresponding to the notes added to the automated measurement report.

## Notes

**Command Syntax:** :QuickMeas:Notes Value [, AllowCoercion]

**Command Argument(s):** Value <string>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Notes Value

**Related Command(s):** Notes?

**Description:** Sets the notes field which is included in the automated measurement report.

Clear Report

## ClearReport

**Command Syntax:** :QuickMeas:ClearReport

**Command Argument(s):** None

**Example:** :QuickMeas:ClearReport

**Description:** Clears the contents of the automated measurement report.

Autom. Meas.

## ExecAutomMeas

**Command Syntax:** :QuickMeas:ExecAutomMeas

**Command Argument(s):** None

**Example:** :QuickMeas:ExecAutomMeas

**Description:** Begins execution of the automated measurements.

## CancelAutomMeas

**Command Syntax:** :QuickMeas:CancelAutomMeas

**Command Argument(s):** None

**Example:** :QuickMeas:CancelAutomMeas

**Description:** Cancels execution of the automated measurement set.

## OpenReport

*Command Syntax:* :QuickMeas:OpenReport

*Command Argument(s):* None

*Example:* :QuickMeas:OpenReport

*Description:* Opens the automated measurement report.

## CloseReport

*Command Syntax:* :QuickMeas:CloseReport

*Command Argument(s):* None

*Example:* :QuickMeas:CloseReport

*Description:* Closes the automated measurement report.

## ExportReportPDF

*Command Syntax:* :QuickMeas:ExportReportPDF *FileName*

*Command Argument(s):* *FileName* <string>

*Example:* :QuickMeas:ExportReportPDF "MyReport.pdf"

*Description:* Exports the automated measurement report to the named pdf file.

## UpdateReportNotes

*Command Syntax:* :QuickMeas:UpdateReportNotes

*Command Argument(s):* None

*Example:* :QuickMeas:UpdateReportNotes

*Description:* Updates the report to include the current notes string.

## Automated Measurement Configuration Form Commands:

### **OpenAutomMeasForm**

*Command Syntax:* :QuickMeas:OpenAutomMeasForm

*Command Argument(s):* None

*Example:* :QuickMeas:OpenAutomMeasForm

*Description:* Opens an automated measurement form on the current page of the page control.

### **OpenAutomMeasFormwID?**

*Command Syntax:* :QuickMeas:OpenAutomMeasFormwID?

*Command Argument(s):* None

*Response Syntax:* [:QuickMeas:OpenAutomMeasFormwID] FormID

*Response Argument(s):* FormID <int>

*Example:* :QuickMeas:OpenAutomMeasFormwID?  
[:QuickMeas:OpenAutomMeasFormwID] 5

*Description:* Opens an automated measurement form on the current page of the page control and returns the formID of the newly created form.

### **CloseAutomMeasForm**

*Command Syntax:* :QuickMeas:CloseAutomMeasForm FormID

*Command Argument(s):* FormID <int>

*Example:* :QuickMeas:CloseAutomMeasForm 5

*Description:* Closes the automated measurement form with the specified formID.

### **CloseAutomMeasForms**

*Command Syntax:* :QuickMeas:CloseAutomMeasForms

*Command Argument(s):* None

*Example:* :QuickMeas:CloseAutomMeasForms

*Description:* Closes all automated measurement forms on all pages of the page control.



## AutomMeasFormCount?

*Command Syntax:* :QuickMeas:AutomMeasFormCount?

*Command Argument(s):* None

*Response Syntax:* [:QuickMeas:AutomMeasFormCount] Count

*Response Argument(s):* Count <int>

*Example:* :QuickMeas:AutomMeasFormCount?  
[:QuickMeas:AutomMeasFormCount] 2

*Description:* Returns the total number of open automated measurement forms on all pages of the page control.

## AutomMeasFormID?

*Command Syntax:* :QuickMeas:AutomMeasFormID? Index

*Command Argument(s):* Index <int>

*Response Syntax:* [:QuickMeas:AutomMeasFormID] FormID

*Response Argument(s):* FormID <int>

*Example:* :QuickMeas:AutomMeasFormID? Value  
[:QuickMeas:AutomMeasFormID] Value

*Description:* Returns the formID of the Index<sup>th</sup> automated measurement form.

## Supported Form Commands:

**:QuickMeas:OpenForm**  
**:QuickMeas:OpenFormwID?**  
**:QuickMeas:CloseForm**  
**:QuickMeas:CloseForms**  
**:QuickMeas:FormCount?**  
**:QuickMeas:FormID?**

## 2.3.21.1 Level

<b>Object:</b>	<b>:QuickMeas:Level</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands relating to the quick measurements level measurement.



## Waveform?

**Command Syntax:** :QuickMeas:Level:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:Waveform] Value

**Response Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:Level:Waveform?

**[:QuickMeas:Level:Waveform] wsSine**

**Related Command(s):** Waveform

**Description:** Queries the generator waveform used for the level measurement.

## Waveform

**Command Syntax:** :QuickMeas:Level:Waveform Value [, AllowCoercion]

**Command Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the generator waveform used for the level measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:Level:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:FreqStart?  
[:QuickMeas:Level:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the level measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:Level:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the level measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:Level:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:FreqSteps?  
[:QuickMeas:Level:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the level measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:Level:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the level measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:Level:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:FreqStop?  
[:QuickMeas:Level:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the level measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:Level:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the level measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:Level:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:FreqLog?  
[:QuickMeas:Level:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the level measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:Level:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the level measurement frequency sweep.

## LevelStart?

**Command Syntax:** :QuickMeas:Level:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:LevelStart?  
[:QuickMeas:Level:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the level measurement amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:Level:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the level measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:Level:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:LevelSteps?  
[:QuickMeas:Level:LevelSteps] 6

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the level measurement amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:Level:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:LevelSteps 6

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the level measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:Level:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:LevelStop?  
[:QuickMeas:Level:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the level measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:Level:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the level measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:Level:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:LevelLog?  
[:QuickMeas:Level:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the level measurement amplitude sweep.

## LevelLog

**Command Syntax:** :QuickMeas:Level:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the level measurement amplitude sweep.

The screenshot shows a control panel titled "Analysis" with three dropdown menus. The "Highpass" menu is set to "<10 Hz", the "Lowpass" menu is set to "Fs/2", and the "Measure" menu is set to "Level".

## HighpassFilt?

**Command Syntax:** :QuickMeas:Level:HighpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:HighpassFilt] Value

**Response Argument(s):** Value <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

**Example:** :QuickMeas:Level:HighpassFilt?  
[:QuickMeas:Level:HighpassFilt] hpDC

**Related Command(s):** HighpassFilt

**Description:** Queries the highpass filter selection for the level measurement.

## HighpassFilt

**Command Syntax:** :QuickMeas:Level:HighpassFilt Value [, AllowCoercion]

**Command Argument(s):** Value <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:HighpassFilt hpDC

**Related Command(s):** HighpassFilt?

**Description:** Sets the highpass filter selection for the level measurement.

## LowpassFilt?

**Command Syntax:** :QuickMeas:Level:LowpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:LowpassFilt] Value

**Response Argument(s):** Value <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

**Example:** :QuickMeas:Level:LowpassFilt?  
[:QuickMeas:Level:LowpassFilt] lpFsdiv2

**Related Command(s):** LowpassFilt

**Description:** Queries the lowpass filter selection for the level measurement.

## LowpassFilt

**Command Syntax:** :QuickMeas:Level:LowpassFilt Value [, AllowCoercion]

**Command Argument(s):** Value <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:LowpassFilt lpFsdiv2

**Related Command(s):** LowpassFilt?

**Description:** Sets the lowpass filter selection for the level measurement.

## Meas?

**Command Syntax:** :QuickMeas:Level:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:Meas] Value

**Response Argument(s):** Value <int> {mLevel=0 | mRatio=1}

**Example:** :QuickMeas:Level:Meas?

**[ :QuickMeas:Level:Meas ] mLevel**

**Related Command(s):** Meas

**Description:** Queries the level/ratio selection for the level measurement.

## Meas

**Command Syntax:** :QuickMeas:Level:Meas Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mLevel=0 | mRatio=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:Meas mLevel

**Related Command(s):** Meas?

**Description:** Sets the level/ratio selection for the level measurement.

The image shows a software dialog box titled "Settling". It has five input fields arranged in two rows. The first row contains "Profile" (a dropdown menu showing "Exponential") and "Delay" (a text box with "30.000 msec"). The second row contains "Precision" (a text box with "1.0000 %"), "nPoints" (a spinner box with "3"), and "Threshold" (a dropdown menu showing "1.0000 mVrms").

## SettleDelay?

**Command Syntax:** :QuickMeas:Level:SettleDelay? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:QuickMeas:Level:SettleDelay] Value

**Response Argument(s):** Value <unit>

**Example:** :QuickMeas:Level:SettleDelay?

**[ :QuickMeas:Level:SettleDelay ] 0.030 S**

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the level measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.



## SettleDelay

**Command Syntax:** :QuickMeas:Level:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the level measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:Level:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:SettleFloor?  
 [:QuickMeas:Level:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:Level:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:Level:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|  
 stlSequential=4}

**Example:** :QuickMeas:Level:SettleMethod?  
 [:QuickMeas:Level:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:Level:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:Level:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:SettleN?  
 [:QuickMeas:Level:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:Level:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:Level:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Level:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Level:SettleTolerance?  
 [:QuickMeas:Level:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

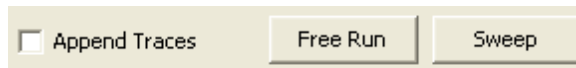
**Command Syntax:** :QuickMeas:Level:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Level:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:Level:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:Level:FreeRun

**Description:** Stars the free-run level measurement.

## Sweep

**Command Syntax:** :QuickMeas:Level:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:Level:Sweep

**Description:** Stars the swept level measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:Level:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Level:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Level:AppendTraces?  
 [:QuickMeas:Level:AppendTraces] **Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:Level:AppendTraces *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Level:AppendTraces *Value*

**Related Command(s):** AppendTraces?

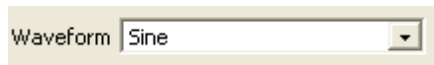
**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

### Supported Form Commands:

**:QuickMeas:Level:OpenForm**  
**:QuickMeas:Level:OpenFormwID?**  
**:QuickMeas:Level:CloseForm**  
**:QuickMeas:Level:CloseForms**  
**:QuickMeas:Level:FormCount?**  
**:QuickMeas:Level:FormID?**

2.3.21.2 Reference

<b>Object:</b>	<b>:QuickMeas:Reference</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurement reference measurement.



### Waveform?

**Command Syntax:** :QuickMeas:Reference:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Reference:Waveform] Value

**Response Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:Reference:Waveform?

**[ :QuickMeas:Reference:Waveform] wsSine**

**Related Command(s):** Waveform

**Description:** Queries the waveform to use for the Reference measurement.

### Waveform

**Command Syntax:** :QuickMeas:Reference:Waveform Value [, AllowCoercion]

**Command Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Reference:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the waveform to use for the Reference measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:Reference:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Reference:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Reference:FreqStart?  
[:QuickMeas:Reference:FreqStart] 1000 HZ

**Related Command(s):** FreqStart

**Description:** Queries the frequency value for the reference measurement.

## FreqStart

**Command Syntax:** :QuickMeas:Reference:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Reference:FreqStart 1000

**Related Command(s):** FreqStart?

**Description:** Sets the frequency value for the reference measurement.

## LevelStart?

**Command Syntax:** :QuickMeas:Reference:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Reference:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Reference:LevelStart?  
[:QuickMeas:Reference:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Queries the frequency at which the reference measurement is made.

## LevelStart

**Command Syntax:** :QuickMeas:Reference:LevelStart *Value* [, *AllowCoercion*]

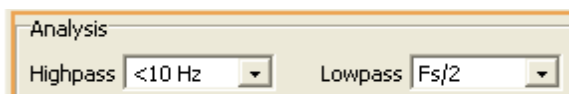
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Reference:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the frequency at which the reference measurement is made.



## HighpassFilt?

**Command Syntax:** :QuickMeas:Reference:HighpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Reference:HighpassFilt] Value

**Response Argument(s):** Value <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

**Example:** :QuickMeas:Reference:HighpassFilt?  
[:QuickMeas:Reference:HighpassFilt] hpDC

**Related Command(s):** HighpassFilt

**Description:** Queries the highpass filter selection for the reference measurement.

## HighpassFilt

**Command Syntax:** :QuickMeas:Reference:HighpassFilt Value [, AllowCoercion]

**Command Argument(s):** Value <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Reference:HighpassFilt hpDC

**Related Command(s):** HighpassFilt?

**Description:** Sets the highpass filter selection for the reference measurement.

## LowpassFilt?

**Command Syntax:** :QuickMeas:Reference:LowpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Reference:LowpassFilt] Value

**Response Argument(s):** Value <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

**Example:** :QuickMeas:Reference:LowpassFilt?  
[:QuickMeas:Reference:LowpassFilt] lpFsdiv2

**Related Command(s):** LowpassFilt

**Description:** Queries the lowpass filter selection for the reference measurement.

## LowpassFilt

**Command Syntax:** :QuickMeas:Reference:LowpassFilt Value [, AllowCoercion]

**Command Argument(s):** Value <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Reference:LowpassFilt lpFsdiv2

**Related Command(s):** LowpassFilt?

**Description:** Sets the lowpass filter selection for the reference measurement.

## FreeRun

*Command Syntax:* :QuickMeas:Reference:FreeRun

*Command Argument(s):* None

*Example:* :QuickMeas:Reference:FreeRun

*Description:* Starts the free run reference measurement.

### Supported Form Commands:

**:QuickMeas:Reference:OpenForm**  
**:QuickMeas:Reference:OpenFormwID?**  
**:QuickMeas:Reference:CloseForm**  
**:QuickMeas:Reference:CloseForms**  
**:QuickMeas:Reference:FormCount?**  
**:QuickMeas:Reference:FormID?**



2.3.21.3 SNR

<b>Object:</b>	<b>:QuickMeas:SNR</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements Signal-to-Noise (SNR) measurement.



### Waveform?

**Command Syntax:** :QuickMeas:SNR:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:Waveform] Value

**Response Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:SNR:Waveform?

**[ :QuickMeas:SNR:Waveform] wsSine**

**Related Command(s):** Waveform

**Description:** Queries the generator waveform used for the SNR measurement.

### Waveform

**Command Syntax:** :QuickMeas:SNR:Waveform Value [, AllowCoercion]

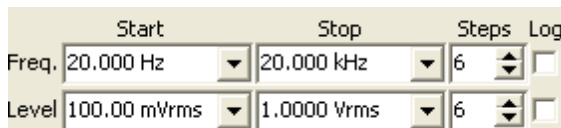
**Command Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the generator waveform used for the SNR measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:SNR:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:FreqStart?  
[:QuickMeas:SNR:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the SNR measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:SNR:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the SNR measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:SNR:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:FreqSteps?  
[:QuickMeas:SNR:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the SNR measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:SNR:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the SNR measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:SNR:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:FreqStop?  
[:QuickMeas:SNR:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the SNR measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:SNR:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the SNR measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:SNR:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:FreqLog?  
[:QuickMeas:SNR:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the SNR measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:SNR:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the SNR measurement frequency sweep.

## LevelStart?

**Command Syntax:** :QuickMeas:SNR:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:LevelStart?  
[:QuickMeas:SNR:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the SNR measurement amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:SNR:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the SNR measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:SNR:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:LevelSteps?  
[:QuickMeas:SNR:LevelSteps] 6

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the SNR measurement amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:SNR:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:LevelSteps 6

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the SNR measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:SNR:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:LevelStop?  
[:QuickMeas:SNR:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the SNR measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:SNR:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the SNR measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:SNR:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:LevelLog?  
[:QuickMeas:SNR:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the SNR measurement amplitude sweep.

## LevelLog

**Command Syntax:** :QuickMeas:SNR:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the SNR measurement amplitude sweep.

Weighting

## WeightingFilt?

**Command Syntax:** :QuickMeas:SNR:WeightingFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:WeightingFilt] *Value*

**Response Argument(s):** *Value* <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

**Example:** :QuickMeas:SNR:WeightingFilt?  
[:QuickMeas:SNR:WeightingFilt] adNoWt

**Related Command(s):** WeightingFilt

**Description:** Queries the weighting filter selection for the SNR measurement.

## WeightingFilt

**Command Syntax:** :QuickMeas:SNR:WeightingFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:WeightingFilt adNoWt

**Related Command(s):** WeightingFilt?

**Description:** Sets the weighting filter selection for the SNR measurement.

The image shows a 'Settling' dialog box with the following settings:

- Profile: Exponential
- Delay: 30.000 msec
- Precision: 1.0000 %
- nPoints: 3
- Threshold: 1.0000 mVrms

## SettleDelay?

**Command Syntax:** :QuickMeas:SNR:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:SettleDelay?  
[:QuickMeas:SNR:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the SNR measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:SNR:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the SNR measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:SNR:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:SettleFloor?  
 [:QuickMeas:SNR:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:SNR:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:SNR:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:SNR:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|  
 stlSequential=4}

**Example:** :QuickMeas:SNR:SettleMethod?  
 [:QuickMeas:SNR:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:SNR:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:SNR:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:SettleN?  
 [:QuickMeas:SNR:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:SNR:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:SNR:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:SNR:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:SNR:SettleTolerance?  
 [:QuickMeas:SNR:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.



## SettleTolerance

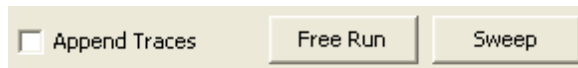
**Command Syntax:** :QuickMeas:SNR:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:SNR:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:SNR:FreeRun

**Description:** Stars the free-run SNR measurement.

## Sweep

**Command Syntax:** :QuickMeas:SNR:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:SNR:Sweep

**Description:** Stars the swept SNR measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:SNR:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:SNR:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:SNR:AppendTraces?  
 [:QuickMeas:SNR:AppendTraces] 0

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces (1) to the graph or replace existing traces (0).

## AppendTraces

**Command Syntax:** :QuickMeas:SNR:AppendTraces *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:SNR:AppendTraces *Value*

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

### *Supported Form Commands:*

**:QuickMeas:SNR:OpenForm**  
**:QuickMeas:SNR:OpenFormwID?**  
**:QuickMeas:SNR:CloseForm**  
**:QuickMeas:SNR:CloseForms**  
**:QuickMeas:SNR:FormCount?**  
**:QuickMeas:SNR:FormID?**

2.3.21.4 THD+N

<b>Object:</b>	<b>:QuickMeas:THDN</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements THD+N measurement.



### Waveform?

**Command Syntax:** :QuickMeas:THDN:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:Waveform] Value

**Response Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:THDN:Waveform?  
[:QuickMeas:THDN:Waveform] wsSine

**Related Command(s):** Waveform

**Description:** Queries the generator waveform used for the THD+N measurement.

### Waveform

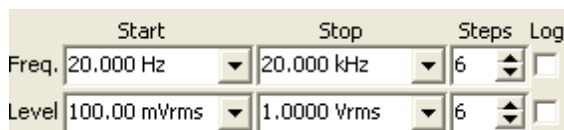
**Command Syntax:** :QuickMeas:THDN:Waveform Value [, AllowCoercion]

**Command Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the generator waveform used for the THD+N measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:THDN:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:FreqStart?  
[:QuickMeas:THDN:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the THD+N measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:THDN:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the THD+N measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:THDN:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:FreqSteps?  
[:QuickMeas:THDN:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the THD+N measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:THDN:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the THD+N measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:THDN:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:FreqStop?  
[:QuickMeas:THDN:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the THD+N measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:THDN:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the THD+N measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:THDN:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:FreqLog?  
[:QuickMeas:THDN:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the THD+N measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:THDN:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the THD+N measurement frequency sweep.

## LevelStart?

**Command Syntax:** :QuickMeas:THDN:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:LevelStart?  
[:QuickMeas:THDN:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the THD+N measurement amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:THDN:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the THD+N measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:THDN:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:LevelSteps?  
[:QuickMeas:THDN:LevelSteps] 6

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the THD+N measurement amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:THDN:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:LevelSteps 6

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the THD+N measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:THDN:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:LevelStop?  
[:QuickMeas:THDN:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the THD+N measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:THDN:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the THD+N measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:THDN:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:LevelLog?  
[:QuickMeas:THDN:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the THD+N measurement amplitude sweep.

## LevelLog

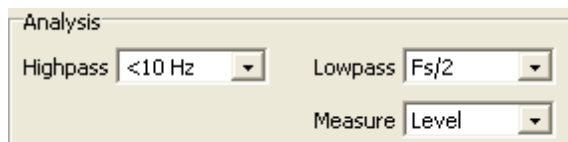
**Command Syntax:** :QuickMeas:THDN:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the THD+N measurement amplitude sweep.



## HighpassFilt?

**Command Syntax:** :QuickMeas:THDN:HighpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:HighpassFilt] *Value*

**Response Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

**Example:** :QuickMeas:THDN:HighpassFilt?  
[:QuickMeas:THDN:HighpassFilt] hpDC

**Related Command(s):** HighpassFilt

**Description:** Queries the highpass filter selection for the THD+N measurement.

## HighpassFilt

**Command Syntax:** :QuickMeas:THDN:HighpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:HighpassFilt hpDC

**Related Command(s):** HighpassFilt?

**Description:** Sets the highpass filter selection for the THD+N measurement.

## LowpassFilt?

**Command Syntax:** :QuickMeas:THDN:LowpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:LowpassFilt] *Value*

**Response Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

**Example:** :QuickMeas:THDN:LowpassFilt?  
[:QuickMeas:THDN:LowpassFilt] lpFsdiv2

**Related Command(s):** LowpassFilt

**Description:** Queries the lowpass filter selection for the THD+N measurement.

## LowpassFilt

**Command Syntax:** :QuickMeas:THDN:LowpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:LowpassFilt lpFsdiv2

**Related Command(s):** LowpassFilt?

**Description:** Sets the lowpass filter selection for the THD+N measurement.



## Meas?

**Command Syntax:** :QuickMeas:THDN:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:Meas] Value

**Response Argument(s):** Value <int> {mLevel=0 | mRatio=1}

**Example:** :QuickMeas:THDN:Meas?

**[ :QuickMeas:THDN:Meas ] mLevel**

**Related Command(s):** Meas

**Description:** Queries the level/ratio selection for the THD+N measurement.

## Meas

**Command Syntax:** :QuickMeas:THDN:Meas Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mLevel=0 | mRatio=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:Meas mLevel

**Related Command(s):** Meas?

**Description:** Sets the level/ratio selection for the THD+N measurement.

## SettleDelay?

**Command Syntax:** :QuickMeas:THDN:SettleDelay? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:QuickMeas:THDN:SettleDelay] Value

**Response Argument(s):** Value <unit>

**Example:** :QuickMeas:THDN:SettleDelay?

**[ :QuickMeas:THDN:SettleDelay ] 0.030 S**

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the THD+N measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:THDN:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:THDN:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the THD+N measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:THDN:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:SettleFloor?  
 [:QuickMeas:THDN:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:THDN:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:THDN:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:THDN:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|stlSequential=4}

**Example:** :QuickMeas:THDN:SettleMethod?  
 [:QuickMeas:THDN:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:THDN:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:THDN:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:SettleN?  
 [:QuickMeas:THDN:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:THDN:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:THDN:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:THDN:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:THDN:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:THDN:SettleTolerance?  
 [:QuickMeas:THDN:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

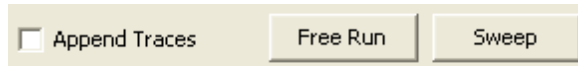
**Command Syntax:** :QuickMeas:THDN:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:THDN:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:THDN:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:THDN:FreeRun

**Description:** Stars the free-run THD+N measurement.

## Sweep

**Command Syntax:** :QuickMeas:THDN:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:THDN:Sweep

**Description:** Stars the swept THD+N measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:THDN:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:THDN:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:THDN:AppendTraces?

**[ :QuickMeas:THDN:AppendTraces ] Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:THDN:AppendTraces *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:THDN:AppendTraces *Value*

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

### Supported Form Commands:

**:QuickMeas:THDN:OpenForm**  
**:QuickMeas:THDN:OpenFormwID?**  
**:QuickMeas:THDN:CloseForm**  
**:QuickMeas:THDN:CloseForms**  
**:QuickMeas:THDN:FormCount?**  
**:QuickMeas:THDN:FormID?**

## 2.3.21.5 Distortion

<b>Object:</b>	<b>:QuickMeas:Distortion</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements distortion measurement.



## Waveform?

**Command Syntax:** :QuickMeas:Distortion:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:Waveform] Value

**Response Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:Distortion:Waveform?

**[ :QuickMeas:Distortion:Waveform] wsSine**

**Related Command(s):** Waveform

**Description:** Queries the generator waveform used for the distortion measurement.

## Waveform

**Command Syntax:** :QuickMeas:Distortion:Waveform Value [, AllowCoercion]

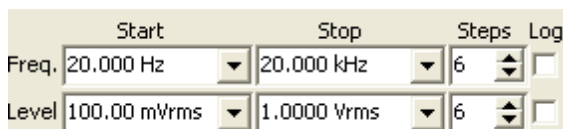
**Command Argument(s):** Value <int> {wsSine=0 | wsLDSine=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the generator waveform used for the distortion measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:Distortion:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:FreqStart?  
[:QuickMeas:Distortion:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the distortion measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:Distortion:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the distortion measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:Distortion:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:FreqSteps?  
[:QuickMeas:Distortion:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the distortion measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:Distortion:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the distortion measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:Distortion:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:FreqStop?  
[:QuickMeas:Distortion:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the distortion measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:Distortion:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the distortion measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:Distortion:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:FreqLog?  
[:QuickMeas:Distortion:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the distortion measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:Distortion:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the distortion measurement frequency sweep.



## LevelStart?

**Command Syntax:** :QuickMeas:Distortion:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:LevelStart?

[ :QuickMeas:Distortion:LevelStart ] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the distortion measurement amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:Distortion:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the distortion measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:Distortion:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:LevelSteps?

[ :QuickMeas:Distortion:LevelSteps ] 6

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the distortion measurement amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:Distortion:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:LevelSteps 6

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the distortion measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:Distortion:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:LevelStop?  
[:QuickMeas:Distortion:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the distortion measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:Distortion:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the distortion measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:Distortion:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:LevelLog?  
[:QuickMeas:Distortion:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the distortion measurement amplitude sweep.

## LevelLog

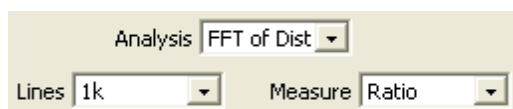
**Command Syntax:** :QuickMeas:Distortion:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the distortion measurement amplitude sweep.



## Analysis?

**Command Syntax:** :QuickMeas:Distortion:Analysis?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:Analysis] Value

**Response Argument(s):** Value <int> {daTHD=0 | daHarmonics=1 | daFFTofHarmonics=2}

**Example:** :QuickMeas:Distortion:Analysis?

**[:QuickMeas:Distortion:Analysis] daTHD**

**Related Command(s):** Analysis

**Description:** Queries the analysis method for the distortion measurement.

## Analysis

**Command Syntax:** :QuickMeas:Distortion:Analysis Value [, AllowCoercion]

**Command Argument(s):** Value <int> {daTHD=0 | daHarmonics=1 | daFFTofHarmonics=2}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:Analysis daTHD

**Related Command(s):** Analysis?

**Description:** Sets the analysis method for the distortion measurement.

## Meas?

**Command Syntax:** :QuickMeas:Distortion:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:Meas] Value

**Response Argument(s):** Value <int> {mLevel=0 | mRatio=1}

**Example:** :QuickMeas:Distortion:Meas?

**[:QuickMeas:Distortion:Meas] mLevel**

**Related Command(s):** Meas

**Description:** Queries the level/ratio setting for the distortion measurement.

## Meas

**Command Syntax:** :QuickMeas:Distortion:Meas Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mLevel=0 | mRatio=1}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:Meas mLevel

**Related Command(s):** Meas?

**Description:** Sets the level/ratio setting for the distortion measurement.

## FFTLines?

**Command Syntax:** :QuickMeas:Distortion:FFTLines?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:FFTLines] Value

**Response Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

**Example:** :QuickMeas:Distortion:FFTLines?

**[ :QuickMeas:Distortion:FFTLines ] fft132k**

**Related Command(s):** FFTLines

**Description:** Queries the FFT resolution (number of lines) for the FFT-based distortion measurement.

## FFTLines

**Command Syntax:** :QuickMeas:Distortion:FFTLines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:FFTLines fft132k

**Related Command(s):** FFTLines?

**Description:** Sets the FFT resolution (number of lines) for the FFT-based distortion measurement.

The image shows a software dialog box titled "Settling". It has several input fields and dropdown menus. The "Profile" dropdown is set to "Exponential". The "Delay" text box contains "30.000 msec". The "Precision" text box contains "1.0000 %". The "nPoints" spinner box is set to "3". The "Threshold" dropdown is set to "1.0000 mVrms".

## SettleDelay?

**Command Syntax:** :QuickMeas:Distortion:SettleDelay? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:SettleDelay] Value

**Response Argument(s):** Value <unit>

**Example:** :QuickMeas:Distortion:SettleDelay?

**[ :QuickMeas:Distortion:SettleDelay ] 0.030 S**

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the distortion measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:Distortion:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the distortion measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:Distortion:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:SettleFloor?  
 [:QuickMeas:Distortion:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:Distortion:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:Distortion:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|  
 stlSequential=4}

**Example:** :QuickMeas:Distortion:SettleMethod?  
 [:QuickMeas:Distortion:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:Distortion:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:Distortion:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:SettleN?  
 [:QuickMeas:Distortion:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:Distortion:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Distortion:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:Distortion:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Distortion:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Distortion:SettleTolerance?  
 [:QuickMeas:Distortion:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

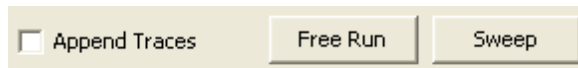
**Command Syntax:** :QuickMeas:Distortion:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:Distortion:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:Distortion:FreeRun

**Description:** Stars the free-run distortion measurement.

## Sweep

**Command Syntax:** :QuickMeas:Distortion:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:Distortion:Sweep

**Description:** Stars the swept distortion measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:Distortion:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Distortion:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Distortion:AppendTraces?  
 [:QuickMeas:Distortion:AppendTraces] **Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:Distortion:AppendTraces *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Distortion:AppendTraces *Value*

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

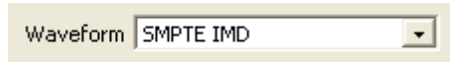
### Supported Form Commands:

**:QuickMeas:Distortion:OpenForm**  
**:QuickMeas:Distortion:OpenFormwID?**  
**:QuickMeas:Distortion:CloseForm**  
**:QuickMeas:Distortion:CloseForms**  
**:QuickMeas:Distortion:FormCount?**  
**:QuickMeas:Distortion:FormID?**



2.3.21.6 IMD

<b>Object:</b>	<b>:QuickMeas:IMDistortion</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements IMD measurement.



### Waveform?

**Command Syntax:** :QuickMeas:IMDistortion:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:Waveform] Value

**Response Argument(s):** Value <int> {wiSMPTE=0 | wiCCIF=1 | wiDIM=2}

**Example:** :QuickMeas:IMDistortion:Waveform?

[ :QuickMeas:IMDistortion:Waveform] wiSMPTE

**Related Command(s):** Waveform

**Description:** Queries the waveform type used in the IMD measurement.

### Waveform

**Command Syntax:** :QuickMeas:IMDistortion:Waveform Value [, AllowCoercion]

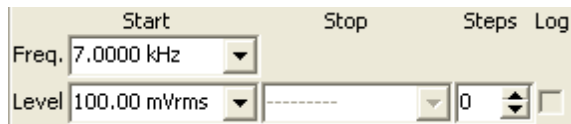
**Command Argument(s):** Value <int> {wiSMPTE=0 | wiCCIF=1 | wiDIM=2}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:Waveform wiSMPTE

**Related Command(s):** Waveform?

**Description:** Sets the waveform type used in the IMD measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:IMDistortion:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:FreqStart?  
[:QuickMeas:IMDistortion:FreqStart] **Value**

**Related Command(s):** FreqStart

**Description:** Queries the main frequency of the IMD waveform used in the measurement.

## FreqStart

**Command Syntax:** :QuickMeas:IMDistortion:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:FreqStart *Value*

**Related Command(s):** FreqStart?

**Description:** Sets the main frequency of the IMD waveform used in the measurement.

## LevelLog?

**Command Syntax:** :QuickMeas:IMDistortion:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:IMDistortion:LevelLog?  
[:QuickMeas:IMDistortion:LevelLog] **0**

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear(0) status of the IMD amplitude sweep.

## LevelLog

**Command Syntax:** :QuickMeas:IMDistortion:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:LevelLog *Value*

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear(0) status of the IMD amplitude sweep.

## LevelStart?

**Command Syntax:** :QuickMeas:IMDistortion:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:LevelStart?  
[:QuickMeas:IMDistortion:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Queries the starting amplitude for the IMD amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:IMDistortion:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:IMDistortion:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the starting amplitude for the IMD amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:IMDistortion:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:IMDistortion:LevelSteps?  
[:QuickMeas:IMDistortion:LevelSteps] 10

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the IMD amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:IMDistortion:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:IMDistortion:LevelSteps 10

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the IMD amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:IMDistortion:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:LevelStop?  
[:QuickMeas:IMDistortion:LevelStop] 1.0 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the IMD amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:IMDistortion:LevelStop *Value* [, *AllowCoercion*]

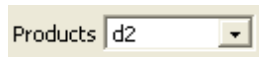
**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:LevelStop *Value*

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the IMD amplitude sweep.



## Product?

**Command Syntax:** :QuickMeas:IMDistortion:Product?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:Product] *Value*

**Response Argument(s):** *Value* <int> {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2}

**Example:** :QuickMeas:IMDistortion:Product?  
[:QuickMeas:IMDistortion:Product] imdSMPTE2

**Related Command(s):** Product

**Description:** Queries the distortion product selection for the IMD measurement.

## Product

**Command Syntax:** :QuickMeas:IMDistortion:Product *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:Product imdSMPTE2

**Related Command(s):** Product?

**Description:** Sets the distortion product selection for the IMD measurement.

## SettleDelay?

**Command Syntax:** :QuickMeas:IMDistortion:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:SettleDelay?  
[:QuickMeas:IMDistortion:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the IMD measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:IMDistortion:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the IMD measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:IMDistortion:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:SettleFloor?  
[:QuickMeas:IMDistortion:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:IMDistortion:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:IMDistortion:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

**Example:** :QuickMeas:IMDistortion:SettleMethod?  
 [:QuickMeas:IMDistortion:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:IMDistortion:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:IMDistortion:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:IMDistortion:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:IMDistortion:SettleN?  
 [:QuickMeas:IMDistortion:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:IMDistortion:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:IMDistortion:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:IMDistortion:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:IMDistortion:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:IMDistortion:SettleTolerance?  
 [:QuickMeas:IMDistortion:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

**Command Syntax:** :QuickMeas:IMDistortion:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:IMDistortion:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:IMDistortion:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:IMDistortion:FreeRun

**Description:** Stars the free-run IMD measurement.

## Sweep

**Command Syntax:** :QuickMeas:IMDistortion:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:IMDistortion:Sweep

**Description:** Stars the swept IMD measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:IMDistortion:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:IMDistortion:AppendTraces] Value

**Response Argument(s):** Value <int>

**Example:** :QuickMeas:IMDistortion:AppendTraces?

**[:QuickMeas:IMDistortion:AppendTraces] Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:IMDistortion:AppendTraces Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0|True=1}

**Example:** :QuickMeas:IMDistortion:AppendTraces Value

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

## Supported Form Commands:

**:QuickMeas:IMDistortion:OpenForm**

**:QuickMeas:IMDistortion:OpenFormID?**

**:QuickMeas:IMDistortion:CloseForm**

**:QuickMeas:IMDistortion:CloseForms**

**:QuickMeas:IMDistortion:FormCount?**

**:QuickMeas:IMDistortion:FormID?**



2.3.21.7 Crosstalk

<b>Object:</b>	<b>:QuickMeas:Crosstalk</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements crosstalk measurement.



### Waveform?

**Command Syntax:** :QuickMeas:Crosstalk:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:Waveform] *Value*

**Response Argument(s):** *Value* <int> {wsSine=0 | wsLDSine=1}

**Example:** :QuickMeas:Crosstalk:Waveform?

**[:QuickMeas:Crosstalk:Waveform] wsSine**

**Related Command(s):** Waveform

**Description:** Queries the generator waveform used for the crosstalk measurement.

### Waveform

**Command Syntax:** :QuickMeas:Crosstalk:Waveform *Value* [, *AllowCoercion*]

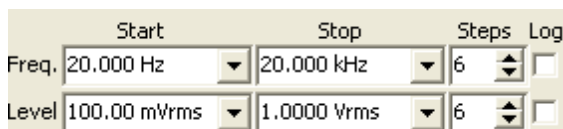
**Command Argument(s):** *Value* <int> {wsSine=0 | wsLDSine=1}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:Waveform wsSine

**Related Command(s):** Waveform?

**Description:** Sets the generator waveform used for the crosstalk measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:Crosstalk:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:FreqStart?  
[:QuickMeas:Crosstalk:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the crosstalk measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:Crosstalk:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the crosstalk measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:Crosstalk:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Crosstalk:FreqSteps?  
[:QuickMeas:Crosstalk:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the crosstalk measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:Crosstalk:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the crosstalk measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:Crosstalk:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:FreqStop?  
[:QuickMeas:Crosstalk:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the crosstalk measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:Crosstalk:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the crosstalk measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:Crosstalk:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Crosstalk:FreqLog?  
[:QuickMeas:Crosstalk:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the crosstalk measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:Crosstalk:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the crosstalk measurement frequency sweep.

## LevelStart?

**Command Syntax:** `:QuickMeas:Crosstalk:LevelStart? [ValueUnit]`

**Command Argument(s):** `ValueUnit <unitstring>`

**Response Syntax:** `[:QuickMeas:Crosstalk:LevelStart] Value`

**Response Argument(s):** `Value <unit>`

**Example:** `:QuickMeas:Crosstalk:LevelStart?  
[:QuickMeas:Crosstalk:LevelStart] 0.1 VRMS`

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the crosstalk measurement amplitude sweep.

## LevelStart

**Command Syntax:** `:QuickMeas:Crosstalk:LevelStart Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`

`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:QuickMeas:Crosstalk:LevelStart 0.1 VRMS`

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the crosstalk measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** `:QuickMeas:Crosstalk:LevelSteps?`

**Command Argument(s):**

**Response Syntax:** `[:QuickMeas:Crosstalk:LevelSteps] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:QuickMeas:Crosstalk:LevelSteps?  
[:QuickMeas:Crosstalk:LevelSteps] 6`

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the crosstalk measurement amplitude sweep.

## LevelSteps

**Command Syntax:** `:QuickMeas:Crosstalk:LevelSteps Value [, AllowCoercion]`

**Command Argument(s):** `Value <int>`

`AllowCoercion <bool> {False=0|True=1}`

**Example:** `:QuickMeas:Crosstalk:LevelSteps 6`

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the crosstalk measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:Crosstalk:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:LevelStop?  
[:QuickMeas:Crosstalk:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the crosstalk measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:Crosstalk:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the crosstalk measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:Crosstalk:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Crosstalk:LevelLog?  
[:QuickMeas:Crosstalk:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the crosstalk measurement amplitude sweep.

## LevelLog

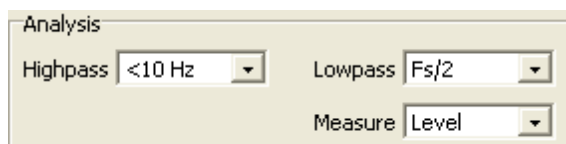
**Command Syntax:** :QuickMeas:Crosstalk:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the crosstalk measurement amplitude sweep.



## HighpassFilt?

**Command Syntax:** :QuickMeas:Crosstalk:HighpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:HighpassFilt] *Value*

**Response Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

**Example:** :QuickMeas:Crosstalk:HighpassFilt?  
[:QuickMeas:Crosstalk:HighpassFilt] hpDC

**Related Command(s):** HighpassFilt

**Description:** Queries the highpass filter selection for the crosstalk measurement.

## HighpassFilt

**Command Syntax:** :QuickMeas:Crosstalk:HighpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:HighpassFilt hpDC

**Related Command(s):** HighpassFilt?

**Description:** Sets the highpass filter selection for the crosstalk measurement.

## LowpassFilt?

**Command Syntax:** :QuickMeas:Crosstalk:LowpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:LowpassFilt] *Value*

**Response Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

**Example:** :QuickMeas:Crosstalk:LowpassFilt?  
[:QuickMeas:Crosstalk:LowpassFilt] lpFsdiv2

**Related Command(s):** LowpassFilt

**Description:** Queries the lowpass filter selection for the crosstalk measurement.

## LowpassFilt

**Command Syntax:** :QuickMeas:Crosstalk:LowpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:LowpassFilt lpFsdiv2

**Related Command(s):** LowpassFilt?

**Description:** Sets the lowpass filter selection for the crosstalk measurement.

## Meas?

**Command Syntax:** :QuickMeas:Crosstalk:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:Meas] *Value*

**Response Argument(s):** *Value* <int> {mLevel=0 | mRatio=1}

**Example:** :QuickMeas:Crosstalk:Meas?  
[:QuickMeas:Crosstalk:Meas] mLevel

**Related Command(s):** Meas

**Description:** Queries the level/ratio selection for the crosstalk measurement.

## Meas

**Command Syntax:** :QuickMeas:Crosstalk:Meas *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {mLevel=0 | mRatio=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:Meas mLevel

**Related Command(s):** Meas?

**Description:** Sets the level/ratio selection for the crosstalk measurement.

## SettleDelay?

**Command Syntax:** :QuickMeas:Crosstalk:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:SettleDelay?  
[:QuickMeas:Crosstalk:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the crosstalk measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:Crosstalk:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the crosstalk measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:Crosstalk:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:SettleFloor?  
 [:QuickMeas:Crosstalk:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:Crosstalk:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:Crosstalk:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:Crosstalk:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|stlSequential=4}

**Example:** :QuickMeas:Crosstalk:SettleMethod?  
 [:QuickMeas:Crosstalk:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.



## SettleMethod

**Command Syntax:** :QuickMeas:Crosstalk:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:Crosstalk:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Crosstalk:SettleN?  
 [:QuickMeas:Crosstalk:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:Crosstalk:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:Crosstalk:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:Crosstalk:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:Crosstalk:SettleTolerance?  
 [:QuickMeas:Crosstalk:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

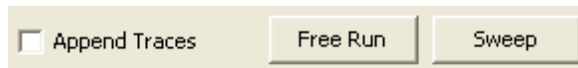
**Command Syntax:** :QuickMeas:Crosstalk:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:Crosstalk:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:Crosstalk:FreeRun

**Description:** Stars the free-run crosstalk measurement.

## Sweep

**Command Syntax:** :QuickMeas:Crosstalk:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:Crosstalk:Sweep

**Description:** Stars the swept crosstalk measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:Crosstalk:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:Crosstalk:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:Crosstalk:AppendTraces?  
 [:QuickMeas:Crosstalk:AppendTraces] *Value*

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:Crosstalk:AppendTraces *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:Crosstalk:AppendTraces *Value*

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

### Supported Form Commands:

**:QuickMeas:Crosstalk:OpenForm**  
**:QuickMeas:Crosstalk:OpenFormwID?**  
**:QuickMeas:Crosstalk:CloseForm**  
**:QuickMeas:Crosstalk:CloseForms**  
**:QuickMeas:Crosstalk:FormCount?**  
**:QuickMeas:Crosstalk:FormID?**

## 2.3.21.8 Frequency Response

<b>Object:</b>	<b>:QuickMeas:FreqResp</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements Frequency Response measurement.



## Waveform?

**Command Syntax:** :QuickMeas:FreqResp:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:Waveform] Value

**Response Argument(s):** Value <int> {wcChirp=0}

**Example:** :QuickMeas:FreqResp:Waveform?

[ :QuickMeas:FreqResp:Waveform] wcChirp

**Related Command(s):** Waveform

**Description:** Queries the waveform used in the frequency response measurement.

## Waveform

**Command Syntax:** :QuickMeas:FreqResp:Waveform Value [, AllowCoercion]

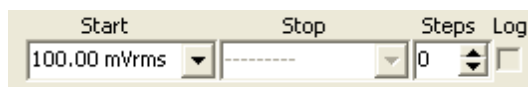
**Command Argument(s):** Value <int> {wcChirp=0}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:Waveform wcChirp

**Related Command(s):** Waveform?

**Description:** Sets the waveform used in the frequency response measurement.



## LevelStart?

**Command Syntax:** :QuickMeas:FreqResp:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:FreqResp:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:FreqResp:LevelStart?  
[:QuickMeas:FreqResp:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Queries the starting amplitude for the amplitude sweep, or the source amplitude for free-run measurements.

## LevelStart

**Command Syntax:** :QuickMeas:FreqResp:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:LevelStart 10

**Related Command(s):** LevelStart?

**Description:** Sets the starting amplitude for the amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:FreqResp:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:FreqResp:LevelSteps?  
[:QuickMeas:FreqResp:LevelSteps] 10

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:FreqResp:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:LevelSteps 10

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:FreqResp:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:FreqResp:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:FreqResp:LevelStop?  
[:QuickMeas:FreqResp:LevelStop] **Value**

**Related Command(s):** LevelStop

**Description:** Queries the stopping level for the amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:FreqResp:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:LevelStop *Value*

**Related Command(s):** LevelStop?

**Description:** Sets the stopping level for the amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:FreqResp:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:FreqResp:LevelLog?  
[:QuickMeas:FreqResp:LevelLog] **0**

**Related Command(s):** LevelLog

**Description:** Queries the log(1)/linear(0) progression of the amplitude sweep.

## LevelLog

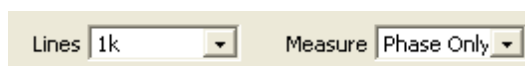
**Command Syntax:** :QuickMeas:FreqResp:LevelLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:LevelLog **1**

**Related Command(s):** LevelLog?

**Description:** Sets the log(1)/linear(0) progression of the amplitude sweep.



## Meas?

**Command Syntax:** :QuickMeas:FreqResp:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:Meas] *Value*

**Response Argument(s):** *Value* <int> {mLevel=0 | mRatio=1}

**Example:** :QuickMeas:FreqResp:Meas?  
[:QuickMeas:FreqResp:Meas] mLevel

**Related Command(s):** Meas

**Description:** Queries the level/ratio selection of the frequency response measurement.

## Meas

**Command Syntax:** :QuickMeas:FreqResp:Meas *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {mLevel=0 | mRatio=1}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:Meas mLevel

**Related Command(s):** Meas?

**Description:** Sets the level/ratio selection of the frequency response measurement.

## FFTLines?

**Command Syntax:** :QuickMeas:FreqResp:FFTLines?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:FFTLines] *Value*

**Response Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 |  
fft1256=7}

**Example:** :QuickMeas:FreqResp:FFTLines?  
[:QuickMeas:FreqResp:FFTLines] fft132k

**Related Command(s):** FFTLines

**Description:** Queries the number of FFT lines (resolution) used in the frequency response measurement.

## FFTLines

**Command Syntax:** :QuickMeas:FreqResp:FFTLines *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 |  
fft1256=7}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:FFTLines fft132k

**Related Command(s):** FFTLines?

**Description:** Sets the number of FFT lines (resolution) used in the frequency response measurement.

The image shows a software dialog box titled "Settling". It has five main input areas:
 

- Profile:** A dropdown menu currently showing "Exponential".
- Delay:** A text input field containing "30.000 msec".
- Precision:** A text input field containing "1.0000 %".
- nPoints:** A spin box currently set to "3".
- Threshold:** A dropdown menu currently showing "1.0000 mVrms".

## SettleDelay?

**Command Syntax:** :QuickMeas:FreqResp:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:FreqResp:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:FreqResp:SettleDelay?  
[:QuickMeas:FreqResp:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the frequency response measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:FreqResp:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the frequency response measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:FreqResp:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:FreqResp:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:FreqResp:SettleFloor?  
[:QuickMeas:FreqResp:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.



## SettleFloor

**Command Syntax:** :QuickMeas:FreqResp:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:FreqResp:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

**Example:** :QuickMeas:FreqResp:SettleMethod?  
 [:QuickMeas:FreqResp:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:FreqResp:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:FreqResp:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:FreqResp:SettleN?  
 [:QuickMeas:FreqResp:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:FreqResp:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:FreqResp:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:FreqResp:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:FreqResp:SettleTolerance?  
 [:QuickMeas:FreqResp:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

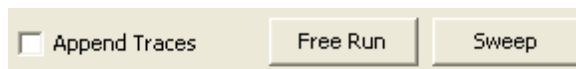
**Command Syntax:** :QuickMeas:FreqResp:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:FreqResp:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:FreqResp:FreeRun

**Description:** Stars the free-run frequency response measurement.

## Sweep

**Command Syntax:** :QuickMeas:FreqResp:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:FreqResp:Sweep

**Description:** Stars the swept frequency response measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:FreqResp:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:FreqResp:AppendTraces] Value

**Response Argument(s):** Value <int>

**Example:** :QuickMeas:FreqResp:AppendTraces?

**[ :QuickMeas:FreqResp:AppendTraces ] Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:FreqResp:AppendTraces Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:FreqResp:AppendTraces Value

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

## Supported Form Commands:

**:QuickMeas:FreqResp:OpenForm**

**:QuickMeas:FreqResp:OpenFormwID?**

**:QuickMeas:FreqResp:CloseForm**

**:QuickMeas:FreqResp:CloseForms**

**:QuickMeas:FreqResp:FormCount?**

**:QuickMeas:FreqResp:FormID?**

## 2.3.21.9 Input/Output Phase

<b>Object:</b>	<b>:QuickMeas:InOutPhase</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurements Input/Output phase measurement.



## Waveform?

**Command Syntax:** :QuickMeas:InOutPhase:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:Waveform] Value

**Response Argument(s):** Value <int> {wcChirp=0}

**Example:** :QuickMeas:InOutPhase:Waveform?

[ :QuickMeas:InOutPhase:Waveform] wcChirp

**Related Command(s):** Waveform

**Description:** Queries the waveform used in the input/output phase measurement.

## Waveform

**Command Syntax:** :QuickMeas:InOutPhase:Waveform Value [, AllowCoercion]

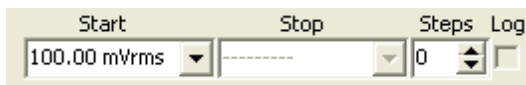
**Command Argument(s):** Value <int> {wcChirp=0}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:Waveform wcChirp

**Related Command(s):** Waveform?

**Description:** Sets the waveform used in the input/output phase measurement.



## LevelStart?

**Command Syntax:** :QuickMeas:InOutPhase:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InOutPhase:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InOutPhase:LevelStart?  
[:QuickMeas:InOutPhase:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Queries the starting amplitude for the amplitude sweep, or the source amplitude for free-run measurements.

## LevelStart

**Command Syntax:** :QuickMeas:InOutPhase:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:LevelStart 10

**Related Command(s):** LevelStart?

**Description:** Sets the starting amplitude for the amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:InOutPhase:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InOutPhase:LevelSteps?  
[:QuickMeas:InOutPhase:LevelSteps] 10

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:InOutPhase:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:LevelSteps 10

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:InOutPhase:LevelStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InOutPhase:LevelStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InOutPhase:LevelStop?  
[:QuickMeas:InOutPhase:LevelStop] **Value**

**Related Command(s):** LevelStop

**Description:** Queries the stopping level for the amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:InOutPhase:LevelStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:LevelStop *Value*

**Related Command(s):** LevelStop?

**Description:** Sets the stopping level for the amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:InOutPhase:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:LevelLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InOutPhase:LevelLog?  
[:QuickMeas:InOutPhase:LevelLog] **0**

**Related Command(s):** LevelLog

**Description:** Queries the log(1)/linear(0) progression of the amplitude sweep.

## LevelLog

**Command Syntax:** :QuickMeas:InOutPhase:LevelLog *Value* [, *AllowCoercion*]

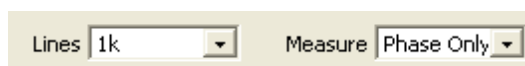
**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:LevelLog **1**

**Related Command(s):** LevelLog?

**Description:** Sets the log(1)/linear(0) progression of the amplitude sweep.



## Meas?

**Command Syntax:** :QuickMeas:InOutPhase:Meas?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:Meas] Value

**Response Argument(s):** Value <int> {mPhase=0 | mPhaseGroupDelay=1}

**Example:** :QuickMeas:InOutPhase:Meas?  
[:QuickMeas:InOutPhase:Meas] mPhase

**Related Command(s):** Meas

**Description:** Queries whether group delay will be included in the measurement.

## Meas

**Command Syntax:** :QuickMeas:InOutPhase:Meas Value [, AllowCoercion]

**Command Argument(s):** Value <int> {mPhase=0 | mPhaseGroupDelay=1}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:Meas mPhase

**Related Command(s):** Meas?

**Description:** Sets whether group delay will be included in the measurement.

## FFTLines?

**Command Syntax:** :QuickMeas:InOutPhase:FFTLines?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:FFTLines] Value

**Response Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 |  
fft1256=7}

**Example:** :QuickMeas:InOutPhase:FFTLines?  
[:QuickMeas:InOutPhase:FFTLines] fft132k

**Related Command(s):** FFTLines

**Description:** Queries the number of FFT lines (resolution) used in the phase measurement.

## FFTLines

**Command Syntax:** :QuickMeas:InOutPhase:FFTLines Value [, AllowCoercion]

**Command Argument(s):** Value <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 |  
fft1256=7}  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:FFTLines fft132k

**Related Command(s):** FFTLines?

**Description:** Sets the number of FFT lines (resolution) used in the phase measurement.

The image shows a software dialog box titled "Settling". It has five main input areas:
 

- Profile:** A dropdown menu showing "Exponential".
- Delay:** A text input field containing "30.000 msec".
- Precision:** A text input field containing "1.0000 %".
- nPoints:** A spin button showing the value "3".
- Threshold:** A dropdown menu showing "1.0000 mVrms".

## SettleDelay?

**Command Syntax:** :QuickMeas:InOutPhase:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InOutPhase:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InOutPhase:SettleDelay?  
[:QuickMeas:InOutPhase:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the phase measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:InOutPhase:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the phase measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:InOutPhase:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InOutPhase:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InOutPhase:SettleFloor?  
[:QuickMeas:InOutPhase:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.



## SettleFloor

**Command Syntax:** `:QuickMeas:InOutPhase:SettleFloor Value [, AllowCoercion]`

**Command Argument(s):** `Value <unit>`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:QuickMeas:InOutPhase:SettleFloor 0.001 VRMS`

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** `:QuickMeas:InOutPhase:SettleMethod?`

**Command Argument(s):**

**Response Syntax:** `[:QuickMeas:InOutPhase:SettleMethod] Value`

**Response Argument(s):** `Value <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}`

**Example:** `:QuickMeas:InOutPhase:SettleMethod?`  
`[:QuickMeas:InOutPhase:SettleMethod] stlFlat`

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** `:QuickMeas:InOutPhase:SettleMethod Value [, AllowCoercion]`

**Command Argument(s):** `Value <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}`  
`AllowCoercion <bool> {False=0 | True=1}`

**Example:** `:QuickMeas:InOutPhase:SettleMethod stlFlat`

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** `:QuickMeas:InOutPhase:SettleN?`

**Command Argument(s):**

**Response Syntax:** `[:QuickMeas:InOutPhase:SettleN] Value`

**Response Argument(s):** `Value <int>`

**Example:** `:QuickMeas:InOutPhase:SettleN?`  
`[:QuickMeas:InOutPhase:SettleN] 3`

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:InOutPhase:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:InOutPhase:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InOutPhase:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InOutPhase:SettleTolerance?  
 [:QuickMeas:InOutPhase:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

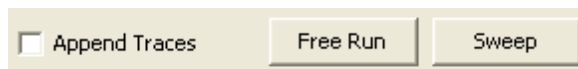
**Command Syntax:** :QuickMeas:InOutPhase:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:InOutPhase:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:InOutPhase:FreeRun

**Description:** Stars the free-run phase measurement.

## Sweep

**Command Syntax:** :QuickMeas:InOutPhase:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:InOutPhase:Sweep

**Description:** Stars the swept phase measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:InOutPhase:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InOutPhase:AppendTraces] Value

**Response Argument(s):** Value <int>

**Example:** :QuickMeas:InOutPhase:AppendTraces?

**[ :QuickMeas:InOutPhase:AppendTraces ] Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:InOutPhase:AppendTraces Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InOutPhase:AppendTraces Value

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

## Supported Form Commands:

**:QuickMeas:InOutPhase:OpenForm**

**:QuickMeas:InOutPhase:OpenFormwID?**

**:QuickMeas:InOutPhase:CloseForm**

**:QuickMeas:InOutPhase:CloseForms**

**:QuickMeas:InOutPhase:FormCount?**

**:QuickMeas:InOutPhase:FormID?**

## 2.3.21.10 InterChannel Phase

<b>Object:</b>	<b>:QuickMeas:InterchPhase</b>
<i>Object Argument(s):</i>	None
<i>Description:</i>	Commands related to the quick measurement interchannel phase measurement.



## Waveform?

**Command Syntax:** :QuickMeas:InterchPhase:Waveform?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:Waveform] Value

**Response Argument(s):** Value <int> {wscSine=0 | wscLDSine=1|wscChirp=2}

**Example:** :QuickMeas:InterchPhase:Waveform?

[ :QuickMeas:InterchPhase:Waveform] wcChirp

**Related Command(s):** Waveform

**Description:** Queries the waveform to use for the Interchannel Phase measurement.

## Waveform

**Command Syntax:** :QuickMeas:InterchPhase:Waveform Value [, AllowCoercion]

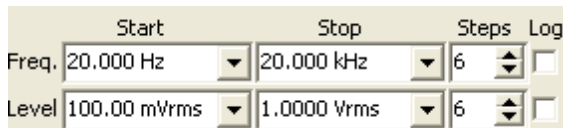
**Command Argument(s):** Value <int> {wcChirp=0}

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:Waveform wcChirp

**Related Command(s):** Waveform?

**Description:** Sets the waveform to use for the Interchannel Phase measurement.



## FreqStart?

**Command Syntax:** :QuickMeas:InterchPhase:FreqStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:FreqStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:FreqStart?  
[:QuickMeas:InterchPhase:FreqStart] 20 HZ

**Related Command(s):** FreqStart

**Description:** Queries the start frequency value for the phase measurement frequency sweep

## FreqStart

**Command Syntax:** :QuickMeas:InterchPhase:FreqStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:FreqStart 20 HZ

**Related Command(s):** FreqStart?

**Description:** Sets the start frequency value for the phase measurement frequency sweep

## FreqSteps?

**Command Syntax:** :QuickMeas:InterchPhase:FreqSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:FreqSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InterchPhase:FreqSteps?  
[:QuickMeas:InterchPhase:FreqSteps] 6

**Related Command(s):** FreqSteps

**Description:** Queries the number of steps in the phase measurement frequency sweep.

## FreqSteps

**Command Syntax:** :QuickMeas:InterchPhase:FreqSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:FreqSteps 6

**Related Command(s):** FreqSteps?

**Description:** Sets the number of steps in the phase measurement frequency sweep..

## FreqStop?

**Command Syntax:** :QuickMeas:InterchPhase:FreqStop? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:FreqStop] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:FreqStop?  
[:QuickMeas:InterchPhase:FreqStop] 20000 HZ

**Related Command(s):** FreqStop

**Description:** Queries the stop frequency for the phase measurement frequency sweep.

## FreqStop

**Command Syntax:** :QuickMeas:InterchPhase:FreqStop *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:InterchPhase:FreqStop 20000

**Related Command(s):** FreqStop?

**Description:** Sets the stop frequency for the phase measurement frequency sweep.

## FreqLog?

**Command Syntax:** :QuickMeas:InterchPhase:FreqLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:FreqLog] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InterchPhase:FreqLog?  
[:QuickMeas:InterchPhase:FreqLog] 0

**Related Command(s):** FreqLog

**Description:** Queries the log (1)/linear (0) status of the phase measurement frequency sweep.

## FreqLog

**Command Syntax:** :QuickMeas:InterchPhase:FreqLog *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:InterchPhase:FreqLog *Value*

**Related Command(s):** FreqLog?

**Description:** Sets the log (1)/linear (0) status of the phase measurement frequency sweep.

## LevelStart?

**Command Syntax:** :QuickMeas:InterchPhase:LevelStart? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:LevelStart] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:LevelStart?  
[:QuickMeas:InterchPhase:LevelStart] 0.1 VRMS

**Related Command(s):** LevelStart

**Description:** Returns the start amplitude for the phase measurement amplitude sweep.

## LevelStart

**Command Syntax:** :QuickMeas:InterchPhase:LevelStart *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:LevelStart 0.1 VRMS

**Related Command(s):** LevelStart?

**Description:** Sets the start amplitude for the phase measurement amplitude sweep.

## LevelSteps?

**Command Syntax:** :QuickMeas:InterchPhase:LevelSteps?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:LevelSteps] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InterchPhase:LevelSteps?  
[:QuickMeas:InterchPhase:LevelSteps] 6

**Related Command(s):** LevelSteps

**Description:** Queries the number of steps in the phase measurement amplitude sweep.

## LevelSteps

**Command Syntax:** :QuickMeas:InterchPhase:LevelSteps *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:LevelSteps 6

**Related Command(s):** LevelSteps?

**Description:** Sets the number of steps in the phase measurement amplitude sweep.

## LevelStop?

**Command Syntax:** :QuickMeas:InterchPhase:LevelStop? [ValueUnit]

**Command Argument(s):** ValueUnit <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:LevelStop] Value

**Response Argument(s):** Value <unit>

**Example:** :QuickMeas:InterchPhase:LevelStop?  
[:QuickMeas:InterchPhase:LevelStop] 1 VRMS

**Related Command(s):** LevelStop

**Description:** Queries the stop amplitude for the phase measurement amplitude sweep.

## LevelStop

**Command Syntax:** :QuickMeas:InterchPhase:LevelStop Value [, AllowCoercion]

**Command Argument(s):** Value <unit>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:LevelStop 1 VRMS

**Related Command(s):** LevelStop?

**Description:** Sets the stop amplitude for the phase measurement amplitude sweep.

## LevelLog?

**Command Syntax:** :QuickMeas:InterchPhase:LevelLog?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:LevelLog] Value

**Response Argument(s):** Value <int>

**Example:** :QuickMeas:InterchPhase:LevelLog?  
[:QuickMeas:InterchPhase:LevelLog] 0

**Related Command(s):** LevelLog

**Description:** Queries the log (1)/linear (0) status of the phase measurement amplitude sweep.

## LevelLog

**Command Syntax:** :QuickMeas:InterchPhase:LevelLog Value [, AllowCoercion]

**Command Argument(s):** Value <int>

AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:LevelLog 0

**Related Command(s):** LevelLog?

**Description:** Sets the log (1)/linear (0) status of the phase measurement amplitude sweep.

The image shows a software control panel titled "Analysis". It contains three dropdown menus: "Highpass" set to "<10 Hz", "Lowpass" set to "Fs/2", and "Lines" set to "-----".



## HighpassFilt?

**Command Syntax:** :QuickMeas:InterchPhase:HighpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:HighpassFilt] *Value*

**Response Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

**Example:** :QuickMeas:InterchPhase:HighpassFilt?  
[:QuickMeas:InterchPhase:HighpassFilt] hpDC

**Related Command(s):** HighpassFilt

**Description:** Queries the highpass filter selection for the phase measurement.

## HighpassFilt

**Command Syntax:** :QuickMeas:InterchPhase:HighpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:HighpassFilt hpDC

**Related Command(s):** HighpassFilt?

**Description:** Sets the highpass filter selection for the phase measurement.

## LowpassFilt?

**Command Syntax:** :QuickMeas:InterchPhase:LowpassFilt?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:LowpassFilt] *Value*

**Response Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

**Example:** :QuickMeas:InterchPhase:LowpassFilt?  
[:QuickMeas:InterchPhase:LowpassFilt] lpFsdiv2

**Related Command(s):** LowpassFilt

**Description:** Queries the lowpass filter selection for the phase measurement.

## LowpassFilt

**Command Syntax:** :QuickMeas:InterchPhase:LowpassFilt *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:LowpassFilt lpFsdiv2

**Related Command(s):** LowpassFilt?

**Description:** Sets the lowpass filter selection for the phase measurement.

## FFTLines?

**Command Syntax:** :QuickMeas:InterchPhase:FFTLines?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:FFTLines] *Value*

**Response Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

**Example:** :QuickMeas:InterchPhase:FFTLines?

[ :QuickMeas:InterchPhase:FFTLines] fft132k

**Related Command(s):** FFTLines

**Description:** Queries the FFT resolution to be used for the measurement.

## FFTLines

**Command Syntax:** :QuickMeas:InterchPhase:FFTLines *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {fft132k=0 | fft116k=1 | fft18k=2 | fft14k=3 | fft12k=4 | fft11k=5 | fft1512=6 | fft1256=7}

*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:FFTLines fft132k

**Related Command(s):** FFTLines?

**Description:** Sets the FFT resolution to be used for the measurement.

The image shows a 'Settling' dialog box with the following controls:

- Profile:** A dropdown menu set to 'Exponential'.
- Delay:** A text input field containing '30.000 msec'.
- Precision:** A text input field containing '1.0000 %'.
- nPoints:** A spin button set to '3'.
- Threshold:** A dropdown menu set to '1.0000 mVrms'.

## SettleDelay?

**Command Syntax:** :QuickMeas:InterchPhase:SettleDelay? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:SettleDelay] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:SettleDelay?

[ :QuickMeas:InterchPhase:SettleDelay] 0.030 S

**Related Command(s):** SettleDelay

**Description:** Queries the settling delay value for the phase measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleDelay

**Command Syntax:** :QuickMeas:InterchPhase:SettleDelay *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:InterchPhase:SettleDelay .003 S

**Related Command(s):** SettleDelay?

**Description:** Sets the settling delay value for the phase measurement. The value is the delay between setting the new sweep value and attempting to obtain a settled measurement.

## SettleFloor?

**Command Syntax:** :QuickMeas:InterchPhase:SettleFloor? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:SettleFloor] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:SettleFloor?  
[:QuickMeas:InterchPhase:SettleFloor] 0.001 VRMS

**Related Command(s):** SettleFloor

**Description:** Queries the smallest value for the settling tolerance window.

## SettleFloor

**Command Syntax:** :QuickMeas:InterchPhase:SettleFloor *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:InterchPhase:SettleFloor 0.001 VRMS

**Related Command(s):** SettleFloor?

**Description:** Sets the smallest value for the settling tolerance window.

## SettleMethod?

**Command Syntax:** :QuickMeas:InterchPhase:SettleMethod?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:SettleMethod] *Value*

**Response Argument(s):** *Value* <int> {stlNone=0|stlExponential=1|stlFlat=2|stlAverage=3|stlSequential=4}

**Example:** :QuickMeas:InterchPhase:SettleMethod?  
[:QuickMeas:InterchPhase:SettleMethod] stlFlat

**Related Command(s):** SettleMethod

**Description:** Queries the settling algorithm.

## SettleMethod

**Command Syntax:** :QuickMeas:InterchPhase:SettleMethod *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int> {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:SettleMethod stlFlat

**Related Command(s):** SettleMethod?

**Description:** Sets the settling algorithm.

## SettleN?

**Command Syntax:** :QuickMeas:InterchPhase:SettleN?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:SettleN] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InterchPhase:SettleN?  
 [:QuickMeas:InterchPhase:SettleN] 3

**Related Command(s):** SettleN

**Description:** Queries the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleN

**Command Syntax:** :QuickMeas:InterchPhase:SettleN *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <int>  
*AllowCoercion* <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:SettleN *Value*

**Related Command(s):** SettleN?

**Description:** Sets the number of measurements that must meet the settling criteria before a measurement is considered settled.

## SettleTolerance?

**Command Syntax:** :QuickMeas:InterchPhase:SettleTolerance? [*ValueUnit*]

**Command Argument(s):** *ValueUnit* <unitstring>

**Response Syntax:** [:QuickMeas:InterchPhase:SettleTolerance] *Value*

**Response Argument(s):** *Value* <unit>

**Example:** :QuickMeas:InterchPhase:SettleTolerance?  
 [:QuickMeas:InterchPhase:SettleTolerance] 1.0 %

**Related Command(s):** SettleTolerance

**Description:** Queries the fractional size of the settling tolerance window.

## SettleTolerance

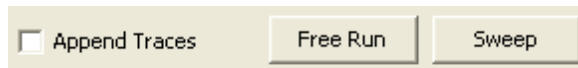
**Command Syntax:** :QuickMeas:InterchPhase:SettleTolerance *Value* [, *AllowCoercion*]

**Command Argument(s):** *Value* <unit>  
*AllowCoercion* <bool> {False=0|True=1}

**Example:** :QuickMeas:InterchPhase:SettleTolerance 10 PCT

**Related Command(s):** SettleTolerance?

**Description:** Sets the fractional size of the settling tolerance window.



## FreeRun

**Command Syntax:** :QuickMeas:InterchPhase:FreeRun

**Command Argument(s):** None

**Example:** :QuickMeas:InterchPhase:FreeRun

**Description:** Stars the free-run phase measurement.

## Sweep

**Command Syntax:** :QuickMeas:InterchPhase:Sweep

**Command Argument(s):** None

**Example:** :QuickMeas:InterchPhase:Sweep

**Description:** Stars the swept phase measurement.

## AppendTraces?

**Command Syntax:** :QuickMeas:InterchPhase:AppendTraces?

**Command Argument(s):**

**Response Syntax:** [:QuickMeas:InterchPhase:AppendTraces] *Value*

**Response Argument(s):** *Value* <int>

**Example:** :QuickMeas:InterchPhase:AppendTraces?  
 [:QuickMeas:InterchPhase:AppendTraces] **Value**

**Related Command(s):** AppendTraces

**Description:** Queries whether new sweeps will append traces to the graph or replace existing traces.

## AppendTraces

**Command Syntax:** :QuickMeas:InterchPhase:AppendTraces Value [, AllowCoercion]

**Command Argument(s):** Value <int>  
AllowCoercion <bool> {False=0 | True=1}

**Example:** :QuickMeas:InterchPhase:AppendTraces Value

**Related Command(s):** AppendTraces?

**Description:** Sets whether new sweeps will append traces to the graph or replace existing traces.

### Supported Form Commands:

**:QuickMeas:InterchPhase:OpenForm**  
**:QuickMeas:InterchPhase:OpenFormwID?**  
**:QuickMeas:InterchPhase:CloseForm**  
**:QuickMeas:InterchPhase:CloseForms**  
**:QuickMeas:InterchPhase:FormCount?**  
**:QuickMeas:InterchPhase:FormID?**

