SR1 Audio Analyzer

SR1 Basic Manual

SRS Stanford Research Systems

Revision 1.0.0
September, 2010

Table of Contents

Introduction

Part I

1 Introduction

The SR1 Basic commands are a collection of properties and functions based on the Microsoft COM binary interface which can be used to program the SR1 Audio Analyzer. Programming may be done on the instrument itself (i.e. locally) using SR1's built-in scripting environment in the VBScript or JScript languages. Programming may also be done on a separate computer (i.e. remotely) using Visual Basic, Microsoft Office, or other COM-enabled software, typically in the VisualBasic language. Remote programming is executed over a computer network, and requires that SR1 and the programming computer have active network connections that allow "DCOM" communication.

This manual is meant to be a complete reference to the syntax of SR1's Scripting command set. This manual is **not** intended to be a reference for VBScript, VisualBasic, or any other language that may be used to program SR1. This manual is also **not** 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 GPIB Interface***, a text-based interface that functions over IEEE 488.2, serial port, or VXI-11 (TCP/IP). The SR1 GPIB interface is documented separately in the SR1 GPIB Programming Manual.

1.1 Revision History

Revision	Date	Comments
1.0	9/10	First complete version

SR1 Basic

Part II

SRS

Audio

2 SR1 Basic

2.1 Using SR1 Basic Commands

SR1 Type Library

All the COM interfaces exposed by SR1 can be found in the Application **SR1.exe**, or the SR1 Type Library **SR1.tlb**. These interfaces may be browsed using, for example, Visual Basic. In addition, the local scripting environment on SR1 has a help panel that lists all the commands in the SR1 Type Library.

There are differences in language conventions between VisualBasic, VBScript, JScript, and other languages that support COM objects. This manual will largely focus of VBScript, which is used in the local SR1 scripting environment, and VisualBasic, which is found in Visual Basic and Microsoft Office environments.

Root & Events CoClasses

There are two CoClasses in SR1 through which all commands are derived. They are **Root**, and **Events**. The Root CoClass is the root of all user commands to SR1. The Events CoClass is the source of all event-based commands from SR1, and is the means by which SR1 notifies the user of events. In the local SR1 scripting environment, instances of these CoClasses have already been created; the instance of Root is "SR1," and the instance of Events is "Events." All commands described in this manual assume these instance names.

If interfacing with SR1 remotely (via Visual Basic for example), these instances need to be created before you can execute any commands. This can be done using the following lines of VisualBasic code:

```
Dim SR1 As SR1.Root
Dim Events As SR1.Events
SR1 = CreateObject("SR1.Root", hostname)
Events = CreateObject("SR1.Events", hostname)
```

where `hostname` is the network name of the SR1 instrument you wish to communicate with (e.g. "\\SR1-1001"). Note that remote COM programming uses the network (typically TCP/IP) to communicate with SR1, and requires certain network ports to be open on both SR1 and the controlling computer. Please see your System Administrator to enable the appropriate ports.

Objects

Each of SR1's Scripting commands belongs to a particular ***Object***. Each Object (COM Interface) 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 object which describes a sine waveform on channel A of the analog generator would be:

```
SR1.AnlgGen.AGenChA.aSine(0)
```

Each object in the string is separated by the period character (.). The SR1 at the beginning of the object is the instance of the SR1 Root CoClass. The first object referenced after that is the Analog Generator, followed by the A channel of the analog generator. The A channel of the analog generator can contain up to 4 waveforms of different types. The final segment of the example object references the first sine waveform belonging to 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). If this argument is not specified, 0 is assumed.

Objects may be assigned to variables. The following VBScript code:

```
Set Sine0 = SR1.AnlgGen.AGenChA.aSine(0)
```

assigns Sine0 to the first analog sine object of analog generator channel A. Using Sine0 instead of SR1.AnlgGen.AGenChA.aSine(0) will save on retyping long object names. VisualBasic is similar, except that the "Set" keyword is not necessary.

Object Properties

Many objects have properties that can be read or set. For instance

```
SR1.Sweep.Source(0).IntNumSteps = 100
```

instructs SR1 to set the number of steps for sweep source 0 (the inner sweep) to 100.

The following

```
steps = SR1.Sweep.Source(0).IntNumSteps
msgbox("The number of steps is " & steps)
```

gets the number of steps for the inner sweep, and displays it in a message box.

Object Functions

Objects can also contain functions, which take 0 or more input parameters, and may or may not return values. The following are examples of function calls:

```
Call SR1.Displays.Graph(10).Trace(101).AutoScale()

id = SR1.Displays.Graph(10).AddTrace(1)

Call SR1.Displays.Graph(10).AddTrace(1)
```

The first line autoscales trace with traceID 101 belonging to graph with graphID 10. It has no input parameters nor a return value. The second line adds a new trace to the graph, and returns the traceID of the new trace. The last line is the same as the line before except that we chose to ignore the return value of the function call. In VBScript, calling a function with no return value, or calling a function with a return value that is discarded, requires the use of the "Call" keyword. "Call" is not required in VisualBasic.

Enumerations

Many properties, command arguments, or return values are integers where the integer value is assigned a specific meaning. These values are called enumerations. An example is the analog input high resolution sample rate (HRSR) enumeration:

```
SR1.HRSR
```

that has the following values

```
srHz64k     = 0
srHz128k    = 1
srOSR       = 2
srOSRx2     = 3
```

To set the analog input high resolution sample rate to 128kHz, the following three commands are equivalent.

```
SR1.AnlgInputs.HiResSampleRate = 1

SR1.AnlgInputs.HiResSampleRate = SR1.HRSR.srHz128k    (VisualBasic)

SR1.AnlgInputs.HiResSampleRate = srHz128k             (VBScript)
```

The last command is applicable to the SR1 local scripting environment.

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 Analog Generator Channel object:

AddWaveform

Command Syntax: SR1.AnlgGen.AGenCh[A/B].AddWaveform(*Type*)

Command Argument(s): *Type* as AnlgWaveformType enumeration {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}

Return Value: *ChanID* as Integer

Example: `ChanID = SR1.AnlgGen.AGenChA.AddWaveform(awfSine)`

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:
SR1.AnlgGen.AGenChA.aSine(ChanID)

The top line lists the name of the command. The Command Syntax lists the complete command string including any arguments in the object specifier or in the command. Items in italics such as *Type* represent the name of the argument, not the literal argument. Arguments enclosed in [*square brackets*] represent a choice, so the proper object name above is AGenChA, or AGenChB. The Command Arguments list each argument described in the Command Syntax along with the type of argument. Enumerated arguments may be sent as integer values, or as enumerations. When enumerations are allowed, the possible values for the enumerations along with the equivalent integer values are listed in curly brackets.

Below the argument description is an example of the command is given with all object and command arguments filled in with typical values. Note that the examples do not give every possible legal form of the command or response, just one typical case. 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.

Units and Scripting Commands

Many of the parameters queried and set by Scripting commands have units associated with them, and most can be set in more than one unit. For example

```
val1 = SR1.DigIO.OutputCarrierAmpBal("Vpp")
val2 = SR1.DigIO.OutputCarrierAmpBal("Vp")
```

will result in val1 being twice as large as val2 because val1 is measured peak-to-peak. To get the reading in the current units of a parameter, a blank string may be entered for the unitstring.

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 Scripting commands.

Unit String	***Units***
Amplitude Units	
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.
dBVrms	Decibels relative to 1Vrms.
W	Watts.
FFS	Fraction of Full Scale.
dBFS	Decibels relative to 1 FFS.
%FS pctFS	Percentage of Full Scale.
Phase Units	
deg	Degrees
rad	Radians
Frequency Units	
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.
Other Units	
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)
`%` `pct`	Percent.
`ul`	Unitless unit.

Commands that set the values of unit-ed parameters are of the form:

```
SR1.DigIO.OutputCarrierAmpBal("Vpp") = 2.0
```

Again, to set the value in the current units of the parameter, an empty string may be used.

Although not listed in the command list, the current units for any unit-ed parameter can be queried by sending the command followed by the string "Unit". SR1 responds with a string indicating the current units. For instance:

```
units = SR1.DigIO.OutputCarrierAmpBalUnit
```

will result in `units` having the string "Vpp" if the units are currently Volts peak-to-peak.

2.2 Using Events

Events are means by which the user can be notified synchronously of events. This allows the user to respond to events before allowing SR1 to continue doing what it was doing. Because events occur synchronously, event code should be small and fast, otherwise the normal functioning of SR1 may be impaired.

Events need to be enabled in SR1.EventMgr, and the user must write event handlers in order to respond to events.
There are several events in SR1, but we will take the example of a Sweep Started event.
When a Sweep Started event occurs, SR1 can do several things, including play a tone, log the event to a file, run a script, or fire a COM event.
Firing a COM event is what allows user code to be executed synchronously with the event.
To handle the Sweep Started event, we must enable the Sweep Started event, and then select COM to fire the OnSweepStart event.
In the local SR1 scripting environment, we write an event handler (note the form of the subroutine name):

```
Sub Events_OnSweepStart()
  msgbox("Event handler got called!")
End Sub
```

In VisualBasic, it would be

```
Sub MyEventHandler()
  msgbox("Event handler got called!")
End Sub
AddHandler Events.OnSweepStart, AddressOf MyEventHandler
```

If we run the script containing the event handler, nothing appears to happen. However, if we now start a sweep (manually or through automation), the OnSweepStart event gets called, and a message box appears with the message "Event handler got called!"
We have just inserted our own code into the normal execution of SR1! This enables us to customize the behavior of SR1. Note that SR1 execution is held up until the message box is dismissed. Be careful to keep event handler code short and fast, otherwise SR1 can become very slow.
Also, note that the event handler is resident in memory. Every time you start a sweep, the code in the OnSweepStart handler gets called, even though the script has finished running. To stop script events, press the "Stop" button on the Scripting window (or Call SR1.Scripting.StopEvents()).

Some events have parameters, like the OnSweepStep() event, which passes in an array of doubles representing the sweep coordinate that is currently active:

```
Sub Events_OnSweepStep(SweepCoord)
    ...
End Sub
```

SweepCoord is either an array of size 1 for a 1-dimensional sweep, or size 2 for a 2-dimensional sweep. For a 2-dimensional sweep, the first coordinate is the outer sweep index, and the second is the inner sweep index.

2.3 Some Example Scripts

The following example scripts are included to provide examples of how the SR1 Basic commands can be used by scripts in real-world situations. These scripts can be used on the SR1 instrument or with the demo mode software.

2.3.1 User Interface Script

```
'Demo Script Illustrating Use of Several Scripting Features...


Option Explicit         'Require explicit declaration of variables

dim ge
dim choice, doExit
dim ver


'Display the software version
SR1.Instrument.ActivePage = 1
ver = SR1.Instrument.Version()
MsgBox ver


'Call SR1.Scripting.CloseForm()

'Display the script menu
do

  Call SR1.Scripting.ClearLines()
  Call SR1.Scripting.CloseScriptLogForms()

  choice = SR1.Instrument.UserLaunchChoice("Scripting Demo", _
            "Part 1,Part 2,Part 3,Exit", _
            "Forms and Panels,Pass/Fail Testing,User Interface
Functions" ,50 )

  ' see if it's time to leave...
  doExit = (choice = "Exit") or (choice = "-cancelled-")

  Call SR1.Instrument.Wait(200)

  if not doExit then
     MsgBox("You've chosen: "&choice&vbNewLine _
     &"Press OK when ready")
  end if

  if InStr(choice,"Part 1") then
     call Option1
  end if

  if InStr(choice,"Part 2") then
     call Option2
  end if

  if InStr(choice,"Part 3") then
     call Option3
  end if

loop while not doExit
```

```
SR1.Instrument.ActivePage = 2


'These are the subroutines called by each option.....
Sub Option1

'Option 1 demonstrates operations with forms.....

dim scId


Call SR1.Instrument.LoadPartial("../../gpib/config/0.xml", not fmScripting)

scId = SR1.Scripting.OpenScriptLogFormwID()
Call SR1.Scripting.WriteLine("This is the scripting log
window..."&vbNewLine)
Call SR1.Instrument.Wait(500)
Call SR1.Scripting.WriteLine("The scripting log window can be used to
record"& _
" test results or to communicate instructions to users.")


Call SR1.Scripting.WriteLine(vbNewLine&"Waiting For Test Results:")

dim i,barId

for i = 1 to 5
    Call SR1.Instrument.Wait(500)
    Call SR1.Scripting.WriteLine("@+")    'Use the + suffix to suppress a
linefeed
next

Call SR1.Instrument.Beep()
Call SR1.Scripting.WriteLine("Test Completed!")

'Create a bar display...
barId = SR1.Displays.NewBar()
SR1.Displays.Bar(barId).Maximize = true
SR1.Displays.Bar(barId).Title = "Test Results"
Call SR1.Displays.Bar(barId).WriteMsg("Passed")

Call SR1.Scripting.WriteLine(vbNewLine&"The barchart display can be used "&
_
"to display test results...")
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"Waiting For Test Results:")
for i = 1 to 5
    Call SR1.Instrument.Wait(500)
    Call SR1.Scripting.WriteLine("@+")
next

Call SR1.Instrument.Beep()
Call SR1.Scripting.WriteLine("Test Completed!")
Call SR1.Displays.Bar(barId).WriteMsg("@rFailed")
Call SR1.Scripting.WriteLine(vbNewLine&"And to provide feedback "& _
"to technicians running the test...")

Call SR1.Instrument.Wait(2000)
Call SR1.Displays.Bar(barId).WriteMsg("@yPlease Check Input Attenuator
Section...")
Call SR1.Instrument.Wait(2000)
Call SR1.Displays.Bar(barId).Close()

Call SR1.Scripting.WriteLine("Scripts can Open and Close SR1 Panels....")

dim fftId, agenId,anainId
fftId =  SR1.Alyzr(1).FFT.OpenFormwID()
agenId = SR1.AnlgGen.OpenFormwID()
anainId = SR1.AnlgInputs.OpenFormwID()

Call SR1.Instrument.Wait(2000)
Call SR1.Scripting.WriteLine("Manipulate their Positions and Size....")

'Play with some form positions....
dim posf,posag,posaid
posf = SR1.Instrument.GetFormPos(fftId)   ' the return values are arrays
(x,y)
posag = SR1.Instrument.GetFormPos(agenId)
posaid = SR1.Instrument.GetFormPos(anainId)

posf(1) = posf(1)+100
posag(0) = posag(0)+100
posaid(1)=posaid(1)+100

Call SR1.Instrument.SetFormPos(fftId, posaid)
Call SR1.Instrument.SetFormPos(agenId, posf)
Call SR1.Instrument.SetFormPos(anainId, posag)
Call SR1.Instrument.Wait(2000)

' Change the form state...
Call SR1.Instrument.SetFormState(fftId,fsMinimized)
Call SR1.Instrument.SetFormState(agenId,fsMinimized)
Call SR1.Instrument.SetFormState(anainId,fsMinimized)
Call SR1.Instrument.Wait(2000)

Call SR1.Instrument.SetFormState(fftId,fsNormal)
Call SR1.Instrument.SetFormState(agenId,fsNormal)
Call SR1.Instrument.SetFormState(anainId,fsNormal)


' Manipulate the page control...
Call SR1.Scripting.WriteLine("And the Page Control....")
For i = 1 to 5
    SR1.Instrument.ActivePage = i
    Call SR1.Instrument.MoveForm(scId, i)
    Call SR1.Instrument.Wait(1000)
next

Call SR1.Instrument.MoveForm(scId, 1)
SR1.Instrument.ActivePage = 1

posf(0) = 0
posf(1) = 0

Call SR1.Instrument.SetFormPos(scId, posf)
Call SR1.Instrument.CloseForm(fftId)
Call SR1.Instrument.CloseForm(agenId)
Call SR1.Instrument.CloseForm(anainId)

Call SR1.Instrument.Wait(1000)
Call SR1.Instrument.UserMessage("Returning to Main Menu...", 10)
Call SR1.Scripting.CloseScriptLogForms()

end sub



Sub Option2

' Demonstrate graph limit traces

'Load default configuration
Call SR1.Instrument.LoadPartial("../../gpib/config/0.xml", not fmScripting)

dim scId

scId = SR1.Scripting.OpenScriptLogFormwID()
Call SR1.Scripting.WriteLine("SR1's displays have sophisticated and
flexible "& _
"features for pass-fail testing..."&vbNewLine)
Call SR1.Instrument.Wait(1000)
Call SR1.Scripting.WriteLine("Arbitrary Pass/Fail Limits can be dynamically
created..."&vbNewLine)

dim gId,tId
gId = SR1.Displays.NewGraph()
tId = SR1.Displays.Graph(gId).AddTrace(msA1FFTspectrum)
SR1.Displays.Graph(gId).Trace(tId).Ymax("dBVrms") = 0
Call SR1.Displays.Graph(gId).Trace(tId).AutoScaleY()

Call SR1.Sweep.FreeRun()

dim lu
lu = SR1.Displays.Graph(gId).GraphLimit.NewLimit(lidUpLimit, tId)'Create an
upper limit
Call SR1.Displays.Graph(gId).GraphLimit.EditLimit(lidUpLimit, _
Array(10.0,10000.0,10000.0,15000.0,15000.0,200000.0), "Hz", _
Array(23.0,23.0,   60.0,   60.0,    23.0,   23.0), "dBVrms")         ' and
edit it

dim trc
set trc = SR1.Displays.Graph(gId).Trace(tId)
Call SR1.Displays.Graph(gId).Trace(lu).SetColor(8388863)
SR1.Displays.Graph(gId).GraphLimit.UpLimitEnabled = true

trc.LimitEnabled = true
SR1.Displays.Graph(gId).ApplyAll = 1
Call SR1.Displays.Graph(gId).SelectTrace(trc)
Call SR1.Displays.Graph(gId).AutoScaleY()
Call SR1.Displays.Graph(gId).AutoScaleX()
Call SR1.Instrument.Wait(1000)
Call SR1.Scripting.WriteLine("And using the Event Manager, scripting
subroutines "& _
"can be executed when measurements exceed the limits..." & vbNewLine)


' now we use the event manager to configure and event when the limit test
fails......
Call SR1.Scripting.WriteLine("Waiting for 4 Limit Fails...")
Call SR1.EventMgr.SetEnabled(eGraphDispLimitExceed, true)
Call SR1.EventMgr.SetComEvent(eGraphDispLimitExceed,
ceGraphDispLimitExceed)

' wait for 4 limit fails
do
  Call SR1.Instrument.Wait(500)
loop while   ge<4 ' variable ge is set by event routine

' now disable the limitfail event
Call SR1.EventMgr.SetEnabled(eGraphDispLimitExceed, false)

ge=0
Call SR1.Displays.Graph(gId).Close()

' same idea for the bar graph
Call SR1.Scripting.WriteLine(vbNewLine&"Pass/Fail testing is also available
using the _ Barchart Display..." & vbNewLine)

dim barId
barId = SR1.Displays.NewBar()
SR1.Displays.Bar(barId).AddTrace(msA1FFTspectrum)
SR1.Displays.Bar(barId).Xmin("")=57.0
SR1.Displays.Bar(barId).Xmin("")=58.0
SR1.Displays.Bar(barId).AlarmMax("")=57.3
SR1.Displays.Bar(barId).RangeAlarm = True
SR1.Displays.Bar(barId).AlarmTone = True

Call SR1.EventMgr.SetEnabled(eBarDispLimitExceed, true)
Call SR1.EventMgr.SetComEvent(eBarDispLimitExceed, ceBarDispLimitExceed)

Call SR1.Scripting.WriteLine("Waiting for 4 Limit Fails...")
ge = 0

do
  Call SR1.Instrument.Wait(500)
loop while   ge<4

ge=0
Call SR1.Instrument.Wait(2000)
Call SR1.EventMgr.SetEnabled(eBarDispLimitExceed, false)
SR1.Displays.Bar(barId).Close()

'illustrate limit testing with the eye diagram
Call SR1.Scripting.WriteLine(vbNewLine&"For Eye Diagrams, custom Inner and
Outer _
Limits can be constructed...")
```

```
 dim digId
 digId = SR1.Displays.NewDigitizerDisplay()
 SR1.Displays.DigitizerDisplay(digId).ActiveChart=ddEyeDiagram
 Call SR1.Digitizer.Acquire("")
 Call SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.Enable(True)

 Call SR1.Instrument.Wait(2000)
 Call SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.ChangePoint
(InUp,2, .25, 2)

 Call SR1.Instrument.Wait(2000)
 Call  SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit._
ChangePoint(InUp,2, .25, 3)

 Call SR1.Scripting.WriteLine("Inner Limit Fail Detected!")

 Call SR1.Instrument.Wait(2000)

Call  SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.ChangePoint_
(InUp,2, .25, 2.0)
Call SR1.Instrument.Wait(2000)
dim ol
ol= 3.7

'move the eye limit around
do
  SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.OutLoLimit("")= -ol
  SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.OutUpLimit("")= ol
  ol = ol-.1
  Call SR1.Instrument.Wait(2000)
loop while SR1.Displays.DigitizerDisplay(digId).EyeDiagram.Limit.Exceeded=0

Call SR1.Scripting.WriteLine("Outer Limit Fail Detected!")
Call SR1.Instrument.Wait(2000)
SR1.Displays.DigitizerDisplay(digId).Close()

MsgBox("Returning to Main Menu")

end sub




Sub Option3

'Demonstrate I/O functions of SR1 Scripting

dim scId

Call SR1.Instrument.LoadPartial("../../gpib/config/0.xml", not fmScripting)
scId = SR1.Scripting.OpenScriptLogFormwID()

dim x
Call SR1.Scripting.WriteLine("The SR1 Basic Interface includes several" & _
" I/O functions allowing test scripts to be run without users having to
interact with"_ & " the underlying SR1 interface." & vbNewLine )
Call SR1.Instrument.Wait(2000)
Call SR1.Scripting.WriteLine("These include functions for displaying simple
messages...")
x = SR1.Instrument.UserMessage("User Message", 20)
Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"And getting text input....")
x = SR1.Instrument.UserInput("UserInput",1.5,50)
Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"Functions to select a single item
from a list....")
x = SR1.Instrument.UserChoice("UserChoice..", "A,B,C,D", 50)
Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"Or to select a multiple items....")
x = SR1.Instrument.UserChoiceMulti("UserChoiceMulti..", "A,B,C,D", 50)
Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"The UserLaunchChoice function can
be used " & _
"to implement a main menu....")
 x = SR1.Instrument.UserLaunchChoice("UserLaunchChoice","A,B,C","A is a
Good Choice,B is Better,But C is Best",50)
Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)

Call SR1.Scripting.WriteLine(vbNewLine&"Scripts have access to file
dialogs" & _
"for both loading files....")
 x = SR1.Instrument.UserLoadfile("UserLoadFile",50)
Call SR1.Scripting.WriteLine("The function returned: " & x)

Call SR1.Instrument.Wait(2000)
Call SR1.Scripting.WriteLine(vbNewLine & "and saving files....")
 x = SR1.Instrument.UserSaveFile("UserSaveFile",30)
 Call SR1.Scripting.WriteLine("The function returned: " & x)
Call SR1.Instrument.Wait(2000)


MsgBox("Returning to Main Menu...")
end sub

Sub Events_GraphLimitExceeded(GraphID)
Call SR1.Instrument.Beep()

ge= ge+1
Call SR1.Scripting.WriteLine("Limit Fail Detected at: " & Time)
end sub


'These subroutines get called in response to event manager events
Sub Events_BarLimitExceeded(BarID)
ge= ge+1
```

```
Call SR1.Scripting.WriteLine("Bar Limit Fail Detected at: " & Time)
end sub
```

2.3.2 Frequency Jitter Script

```
'This script measures bobble on SR1's analog frequency counters. The
script
'illustrates how measurement results can be output to files...

' Load default configuration
Call SR1.Instrument.LoadPartial("../../gpib/config/0.xml", not fmScripting)
SR1.Instrument.ActivePage = 1

ver = SR1.Instrument.Version()

' configure analog generator
agenId = SR1.AnlgGen.OpenFormwID()
Call SR1.AnlgGen.AGenChA.ClearWaveforms()
sineSigId = SR1.AnlgGen.AGenChA.AddWaveform(awfSine)  'add a sine signal


' configure analyzers
SR1.Alyzr(0).Function = azTimeDomDet
SR1.Alyzr(1).Function = azTimeDomDet
a0Id = SR1.Alyzr(0).OpenFormwID()
a1Id = SR1.Alyzr(1).OpenFormwID()
SR1.Alyzr(0).Source=azAnlgA
SR1.Alyzr(1).Source=azAnlgB

' configure analog inputs
SR1.AInChA.Source = aiBNC
SR1.AInChB.Source = aiBNC
SR1.AInChA.AutoRange = False
SR1.AInChB.AutoRange = False
SR1.AInChA.Range("Vrms") = 1
SR1.AInChB.Range("Vrms") = 1


' configure new graph
SR1.Instrument.ActivePage = 3
gId = SR1.Displays.NewGraph()
taId = SR1.Displays.Graph(gId).AddTrace(msAnlgFreqA)
tbId = SR1.Displays.Graph(gId).AddTrace(msAnlgFreqB)
SR1.Displays.Graph(gId).ApplyAll  = True

SR1.Displays.Graph(gId).Trace(taId).Ymax("F/R")=1.0001
SR1.Displays.Graph(gId).Trace(taId).Ymin("F/R")=0.9999
SR1.Displays.Graph(gId).Trace(tbId).Ymax("F/R")=1.0001
SR1.Displays.Graph(gId).Trace(tbId).Ymin("F/R")=0.9999
SR1.Displays.Graph(gId).Cursor.Enabled = True
SR1.Displays.Graph(i).Cursor.Independent = True
SR1.Displays.Graph(gId).Cursor.Cursor1Trace =  0
SR1.Displays.Graph(gId).Cursor.Cursor2Trace =  1

'open the script log form
scrId= SR1.Scripting.OpenScriptLogFormwID()


' setup output file

const tristateFalse = 0
```

```
Const FILE_NAME = "FreqJitter.txt"
Set FSO = CreateObject("Scripting.FileSystemObject")
Set objStream = FSO.CreateTextFile(FILE_NAME, True, TristateFalse)


' Now do the test
amp = 0.3

With objStream    'use the file stream we just created...
.WriteLine "Frequency Jitter: " & amp*100 & "% FS"
End With

Call SR1.Scripting.WriteLine("Frequency Jitter: " & amp*100 & "% FS")

'take 20 jitter measurements...
f0 = 20.0
for i = 1 to 50
  df =  getFreqJitter(amp, f0)
  Call SR1.Scripting.WriteLine(f0 & " , " & df)
  With objStream
  .WriteLine f0 & " , " & df
  end With
  f0 = f0 * 1.2
next

'close the file
With objStream
.Close
End With



'the routine for actuall getting the frequency jitter
Function getFreqJitter( amp, freq )
    SR1.AlyzrReferences.AnlgFreq("") = freq
    SR1.AnlgGen.AGenChA.aSine(sineSigId).Amp("") = amp
    SR1.AnlgGen.AGenChA.aSine(sineSigId).Freq("") = freq
    Call SR1.Sweep.FreeRun()
    Call SR1.Instrument.Wait(5000)
    Call SR1.Displays.Graph(gId).AutoScale()
    Call SR1.Instrument.Wait(50000)
    Call SR1.Sweep.Pause()
    Call SR1.Displays.Graph(gId).Cursor.MoveToMax(ciCursor1)
    ymax = SR1.Displays.Graph(gId).Cursor.y1Rdg("")
    Call SR1.Displays.Graph(gId).Cursor.MoveToMin(ciCursor1)
    ymin = SR1.Displays.Graph(gId).Cursor.y1Rdg("")
    delta1 =  ymax-ymin
    Call SR1.Displays.Graph(gId).Cursor.MoveToMax(ciCursor2)
    ymax = SR1.Displays.Graph(gId).Cursor.y2Rdg("")
    Call SR1.Displays.Graph(gId).Cursor.MoveToMin(ciCursor2)
    ymin = SR1.Displays.Graph(gId).Cursor.y2Rdg("")
    delta2 =  ymax-ymin
    getFreqJitter = delta1
    if (delta2>delta1) then getFreqJitter = delta2
end Function
```

2.4 SR1 Basic Command Reference

2.4.1 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: `Call ObjString.OpenForm()`

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

OpenFormwID

Command Syntax: ObjString.OpenFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = ObjString.OpenFormwID()`

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* as Integer

Example: `Call ObjString.CloseForm(11)`

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

CloseForms

Command Syntax: ObjString.CloseForms()

Command Argument(s): None

Example: `Call 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

Return Value: *Count* as Integer

Example: `Count = ObjString.FormCount()`

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* as Integer

Return Value: *FormID* as Integer

Example: `FormID = ObjString.FormID(2)`

Description: Returns the FormID of the Indexth form of the given type. Index = 0 corresponds to the first form.

2.4.2 Analog Inputs

Object:	**SR1.AnlgInputs**
Object Argument(s):	None
Description:	Properties of the Analog Inputs not specific to a particular input channel.

HiResSampleRate

Command Syntax: SR1.AnlgInputs.HiResSampleRate

Command Argument(s): None

Property: HRSR enumeration {srHz64k=0 | srHz128k=1 | srOSR=2 | srOSRx2=3}

Example:

```
val = SR1.AnlgInputs.HiResSampleRate
SR1.AnlgInputs.HiResSampleRate = srHz64k
```

Description: Sample rate enumeration of the Hi-Resolution converter.

Form Commands:

OpenForm

OpenFormwID

CloseForm

CloseForms

FormCount

FormID

2.4.3 Analog Input Channel

Object:	**SR1.AInCh[A/B]**
Object Argument(s):	*ch* as A or B
Description:	Properties of the Analog Inputs specific to a particular input channel.

AutoRange

Command Syntax: SR1.AInCh[A/B].AutoRange

Command Argument(s): None

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AInChB.AutoRange
SR1.AInChB.AutoRange = True
```

Description: AutoRange status for the selected input.

Coupling

Command Syntax: SR1.AInCh[A/B].Coupling

Property: AICPL enumeration {cplAC=0 | cplDC=1}

Example:

```
val = SR1.AInChA.Coupling
SR1.AInChB.Coupling = cplAC
```

Description: Coupling selection for the selected input channel.

OptionalFilter

Command Syntax: SR1.AInCh[A/B].OptionalFilter

Property: infilter enumeration {ifNone=0 | ifFilter1=1 | ifFilter2=2 | ifFilter3=3 | ifFilter4=4 | ifGround=5}

Example:

```
val = SR1.AInChB.OptionalFilter
SR1.AInChA.OptionalFilter = ifNone
```

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

Range

Command Syntax: SR1.AInCh[A/B].Range(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AInChB.Range("Vrms")
SR1.AInChB.Range("Vrms") = 16.00
```

Description: Input range for the selected input channel.

Source

Command Syntax: SR1.AInCh[A/B].Source

Property: AINS enumeration {aiXLR=0 | aiBNC=1 | aiGenMon=2 | aiDigCommonMode=3}

Example:

```
val = SR1.AInChA.Source
SR1.AInChB.Source = aiBNC
```

Description: Input source selection for the selected input channel.

Status

Command Syntax: SR1.AInCh[A/B].Status

Property: AnlgInStat enumeration (read-only) {aiUnderRange=0 | aiInRange=1 | aiOverRange=2}

Example:

```
val = SR1.AInChA.Status
```

Description: Range status for the selected input channel.

Zin

Command Syntax: SR1.AInCh[A/B].Zin

Property: AnlgZin enumeration {aiHiZ=0 | aiz300=1 | aiz600=2}

Example:

```
val = SR1.AInChA.Zin
SR1.AInChA.Zin = aiz300
```

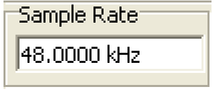
Description: Input impedance selection for the selected channel.

2.4.4 Digital I/O

.

Object:	**SR1.DigIO**
Object Argument(s):	None
Description:	Commands related to features contained on the Digital I/O panel.

Digital Audio Output Commands



Sample Rate

48.0000 kHz

OSR

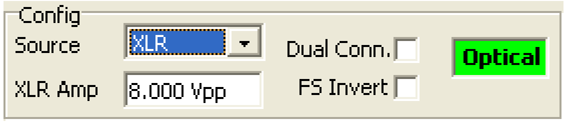
Command Syntax: SR1.DigIO.OSR(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigIO.OSR("")
SR1.DigIO.OSR("") = 48000
```

Description: Digital audio output sampling rate.



Config

Source XLR

XLR Amp 8.000 Vpp

Dual Conn. □

FS Invert □

OutputCarrierAmpBal

Command Syntax: SR1.DigIO.OutputCarrierAmpBal(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigIO.OutputCarrierAmpBal("Vpp")
SR1.DigIO.OutputCarrierAmpBal("Vpp") = 10
```

Description: Amplitude of the balanced digital audio carrier signal.

OutputCarrierAmpUnbal

Command Syntax: SR1.DigIO.OutputCarrierAmpUnbal(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigIO.OutputCarrierAmpUnbal("")
SR1.DigIO.OutputCarrierAmpUnbal("") = 2.0
```

Description: Amplitude of the unbalanced digital audio carrier signal.

OutputDualConnector

Command Syntax: SR1.DigIO.OutputDualConnector

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.OutputDualConnector
SR1.DigIO.OutputDualConnector = Value
```

Description: On/off status of the digital audio output dual-connector setting.

OpticalOutputActive

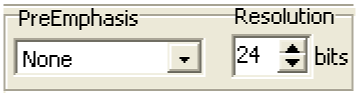
Command Syntax: SR1.DigIO.OpticalOutputActive

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.DigIO.OpticalOutputActive
```

Description: Active status of the digital audio optical output.



PreEmphasis None

Resolution 24 bits

OutputNumBits

Command Syntax: SR1.DigIO.OutputNumBits

Property: Integer

Example:

```
val = SR1.DigIO.OutputNumBits
SR1.DigIO.OutputNumBits = 24
```

Description: Number of bits of output resolution for the digital audio output.

OutputSource

Command Syntax: SR1.DigIO.OutputSource

Property: digOutputSrc enumeration {dosXLR=0 | dosBNC=1}

Example:

```
val = SR1.DigIO.OutputSource
SR1.DigIO.OutputSource = dosXLR
```

Description: Connector for the digital audio output.

Preemphasis

Command Syntax: SR1.DigIO.Preemphasis

Property: DeemphType enumeration {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}

Example:

```
val = SR1.DigIO.Preemphasis
SR1.DigIO.Preemphasis = dioNoDeem
```

Description: Preemphasis mode of the digital audio output.

FrameSyncOutInvert

Command Syntax: SR1.DigIO.FrameSyncOutInvert

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.FrameSyncOutInvert
SR1.DigIO.FrameSyncOutInvert = False
```

Description: Value of the Frame Sync Invert on the Digital Audio Output tab.

Digital Audio Input Commands

ChStatFsARdg

Command Syntax: SR1.DigIO.ChStatFsARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.DigIO.ChStatFsARdg("")`

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

ChStatFsBRdg

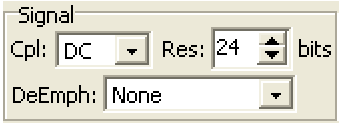
Command Syntax: SR1.DigIO.ChStatFsBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.DigIO.ChStatFsBRdg("")`

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



Cpl: DC
Res: 24 bits
DeEmph: None

Deemphasis

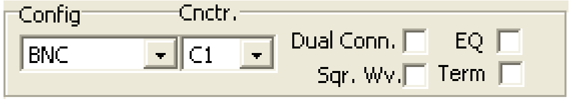
Command Syntax: SR1.DigIO.Deemphasis

Property: DeemphType enumeration {dioNoDeem=0 | dioCD5015dB0=1 | dioCD5015dB10=2 | dioJ17dB0=3 | dioJ17dB20=4}

Example:

```
val = SR1.DigIO.Deemphasis
SR1.DigIO.Deemphasis = dioNoDeem
```

Description Deemphasis applied to the received digital audio signal.



BNC
C1
Dual Conn. □
EQ □
Sqr. Wv. □
Term □

InputBlockDC

Command Syntax: SR1.DigIO.InputBlockDC

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.InputBlockDC
SR1.DigIO.InputBlockDC = False
```

Description: On/off status of the digital audio input DC blocker.

InputCarrierEq

Command Syntax: SR1.DigIO.InputCarrierEq

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.InputCarrierEq
SR1.DigIO.InputCarrierEq = False
```

Description: Status of the digital audio input EQ.

InputConnectorSelect

Command Syntax: SR1.DigIO.InputConnectorSelect

Property: dioConnId enumeration {dioC0=0 | dioC1=1}

Example:
```
val = SR1.DigIO.InputConnectorSelect
SR1.DigIO.InputConnectorSelect = dioC0
```

Description: Digital audio input connector selection.

InputDualConnector

Command Syntax: SR1.DigIO.InputDualConnector

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.InputDualConnector
SR1.DigIO.InputDualConnector = False
```

Description: Status of the digital audio input dual-connector selection.

InputNumBits

Command Syntax: SR1.DigIO.InputNumBits

Property: Integer

Example:
```
val = SR1.DigIO.InputNumBits
SR1.DigIO.InputNumBits = 24
```

Description: Resolution (number of bits) of the digital audio input.

InputSource

Command Syntax: SR1.DigIO.InputSource

Property: DigInSrcType enumeration {diXLR=0 | diBNC=1 | diOptical=2 | diGenMon=3 | diNone=4}

Example:
```
val = SR1.DigIO.InputSource
SR1.DigIO.InputSource = diXLR
```

Description: Digital audio input source selection (XLR, BNC, Optical, or GenMon).

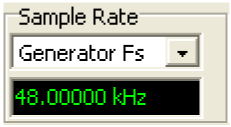
InputTerm

Command Syntax: SR1.DigIO.InputTerm

Property: DigInTerm enumeration {dinHiZ=0 | dinLoZ=1}

Example:
```
val = SR1.DigIO.InputTerm
SR1.DigIO.InputTerm = True
```

Description: Digital audio input termination status.



Sample Rate

Generator Fs

48.00000 kHz

ISRRdg

Command Syntax: SR1.DigIO.ISRRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.DigIO.ISRRdg("48000")
```

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

ISRRef

Command Syntax: SR1.DigIO.ISRRef

Property: DigSampRateSrc enumeration {dioGenerator=0 | dioMeasured=1 | dioStatusBits=2 | dioUser=3}

Example:
```
val = SR1.DigIO.ISRRef
SR1.DigIO.ISRRef = dioGenerator
```

Description: Digital audio input sampling rate mode selection.

UserISRRef

Command Syntax: SR1.DigIO.UserISRRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigIO.UserISRRef("")
SR1.DigIO.UserISRRef("") = 44100
```

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

ReceiverStatusRdg

Command Syntax: SR1.DigIO.ReceiverStatusRdg

Property: RcvrStatusMask enumeration (read-only) {rsmNoErr=0 | rsmParityErr=1 | rsmBiPhaseErr=2 | rsmConfErr=4 | rsmValidityErr=8 | rsmLockErr=16 | rsmCRCErr=32}

Example:
```
val = SR1.DigIO.ReceiverStatusRdg
```

Description: 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

Command Syntax: SR1.DigIO.ReferenceStatusRdg

Property: Integer (read-only)

Example:
```
val = SR1.DigIO.ReferenceStatusRdg
```

Description: 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: SR1.DigIO.SqWaveInput

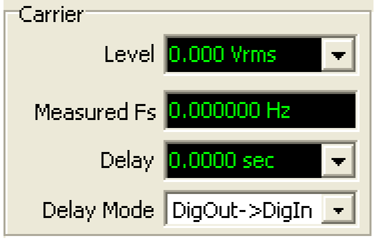
Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.SqWaveInput
SR1.DigIO.SqWaveInput = False
```

Desctipion: 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



Carrier

Level 0.000 Vrms

Measured Fs 0.000000 Hz

Delay 0.0000 sec

Delay Mode DigOut->DigIn

InputCarrierAmpRdg

Command Syntax: SR1.DigIO.InputCarrierAmpRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.DigIO.InputCarrierAmpRdg("")`
Description: Current amplitude measurement of the digital audio input carrier signal.

MeasISRRdg

Command Syntax: SR1.DigIO.MeasISRRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.DigIO.MeasISRRdg("")`
Description: *Measured* value of the digital audio input sampling rate. If the input is set to dual connector this value will refelect the *physical* sampling rate on each wire rather than the combined logical sampling rate.

DelayFromOutRdg

Command Syntax: SR1.DigIO.DelayFromOutRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.DigIO.DelayFromOutRdg("")`
Description: Delay value associated with the current "DelayMode".

DelayMode

Command Syntax: SR1.DigIO.DelayMode
Property: dioDelayMode enumeration {dioDigOut2DigIn=0 | dioRefOut2DigIn=1}
Example:

```
val = SR1.DigIO.DelayMode
SR1.DigIO.DelayMode = 0
```

Description: Value of the Carrier Status "Delay Mode."

HiliteChStatusDiff

Command Syntax: SR1.DigIO.HiliteChStatusDiff

Property: HiliteDiff enumeration {hdNone=0 | hdDiffFromOutput=1 | hdDiffFromOtherCh=2 | hdReservedInUse=3}

Example:

```
val = SR1.DigIO.HiliteChStatusDiff
SR1.DigIO.HiliteChStatusDiff = hdNone
```

Description: Highlight mode used by the channel status and user status forms.

Data/Active Bits Indicator

ActiveBitsA

Command Syntax: SR1.DigIO.ActiveBitsA

Property: Integer (read-only)

Example: `val = SR1.DigIO.ActiveBitsA`

Description: 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: SR1.DigIO.ActiveBitsB

Property: Integer (read-only)

Example: `val = SR1.DigIO.ActiveBitsB`

Description: 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: SR1.DigIO.DataBitsA

Property: Integer (read-only)

Example: `val = SR1.DigIO.DataBitsA`

Description: 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: SR1.DigIO.DataBitsB

Property: Integer (read-only)

Example: `val = SR1.DigIO.DataBitsB`

Description: 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: SR1.DigIO.SetTxChanStat(*Channel*, *Byte*, *Value*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer
Value as Integer

Example: `Call SR1.DigIO.SetTxChanStat(chanA, 0, 4)`

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

SetTxUserStat

Command Syntax: SR1.DigIO.SetTxUserStat(*Channel*, *Byte*, *Value*)

Command Argument (s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer
Value as Integer

Example: `Call SR1.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: SR1.DigIO.GetTxChanStat(*Channel*, *Byte*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer

Return Value: *Value* as Integer

Example: `Value = SR1.DigIO.GetTxChanStat(chanA, 0)`

Description: Value of the transmitted channel status for the channel and byte indicated.

GetTxUserStat

Command Syntax: SR1.DigIO.GetTxUserStat(*Channel*, *Byte*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer

Return Value: *Value* as Integer

Example: `Value = SR1.DigIO.GetTxUserStat(chanA, 0)`

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

TxChStatModeA

Command Syntax: SR1.DigIO.TxChStatModeA

Property: ChStatMode enumeration {csConsumer=0 | csProfessional=1}

Example:
```
val = SR1.DigIO.TxChStatModeA
SR1.DigIO.TxChStatModeA = csConsumer
```

Description: Transmitted Consumer /Professional mode selection for channel A.

TxChStatModeB

Command Syntax: SR1.DigIO.TxChStatModeB

Property: ChStatMode enumeration {csConsumer=0 | csProfessional=1}

Example:
```
val = SR1.DigIO.TxChStatModeB
SR1.DigIO.TxChStatModeB = csProfessional
```

Description: Transmitted Consumer /Professional mode selection for channel B.

TxChStatSelect

Command Syntax: SR1.DigIO.TxChStatSelect

Property: chsel enumeration {chA=0 | chB=1 | chAB=2}

Example:
```
val = SR1.DigIO.TxChStatSelect
SR1.DigIO.TxChStatSelect = chAB
```

Description: 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: SR1.DigIO.GetRxChanStat(*Channel*, *Byte*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer

Return Value: *Value* as Integer

Example:
```
Value = SR1.DigIO.GetRxChanStat(chanA, 1)
```

Description: Indicate byte of the received channel status for the indicate channel.

GetRxUserStat

Command Syntax: SR1.DigIO.GetRxUserStat(*Channel*, *Byte*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}
Byte as Integer

Return Value: *Value* as Integer

Example:
```
Value = SR1.DigIO.GetRxUserStat(chanA, 0)
```

Description: Value of the received user status byte *Byte*.

Channel Status Commands (Professional)

ProAAux

Command Syntax: SR1.DigIO.ProAAux

Property: ProAux enumeration {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}

Example:

```
val = SR1.DigIO.ProAAux
SR1.DigIO.ProAAux = pa20bitNotDef
```

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

ProACh

Command Syntax: SR1.DigIO.ProACh

Property: Integer

Example:

```
val = SR1.DigIO.ProACh
SR1.DigIO.ProACh = 1
```

Description Channel # setting of the channel A transmitted professional channel status.

ProAChMode

Command Syntax: SR1.DigIO.ProAChMode

Property: ProChMode enumeration {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}

Example:

```
val = SR1.DigIO.ProAChMode
SR1.DigIO.ProAChMode = pcmMono
```

Description: Channel mode for the channel A transmitted professional channel status.

ProAConf0

Command Syntax: SR1.DigIO.ProAConf0

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProAConf0
SR1.DigIO.ProAConf0 = True
```

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

ProAConf1

Command Syntax: SR1.DigIO.ProAConf1

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProAConf1
SR1.DigIO.ProAConf1 = False
```

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

ProAConf2

Command Syntax: SR1.DigIO.ProAConf2

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProAConf2
SR1.DigIO.ProAConf2 = True
```

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

ProAConf3

Command Syntax: SR1.DigIO.ProAConf3

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProAConf3
SR1.DigIO.ProAConf3 = False
```

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

ProACrc

Command Syntax: SR1.DigIO.ProACrc

Property: ProCRC enumeration {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}

Example:
```
val = SR1.DigIO.ProACrc
SR1.DigIO.ProACrc = pcCorrect
```

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

ProAData

Command Syntax: SR1.DigIO.ProAData

Property: DigitalData enumeration {ddPCM=0 | ddData=1}

Example:
```
val = SR1.DigIO.ProAData
SR1.DigIO.ProAData = ddPCM
```

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

ProADest

Command Syntax: SR1.DigIO.ProADest

Property: String

Example:
```
val = SR1.DigIO.ProADest
SR1.DigIO.ProADest = srs1
```

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

ProAEmph

Command Syntax: SR1.DigIO.ProAEmph

Property: ProEmph enumeration {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}

Example:

```
val = SR1.DigIO.ProAEmph
SR1.DigIO.ProAEmph = peNone
```

Description: Emphasis mode selection for the channel A transmitted professional channel status.

ProAFs

Command Syntax: SR1.DigIO.ProAFs

Property: ProFs enumeration {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}

Example:

```
val = SR1.DigIO.ProAFs
SR1.DigIO.ProAFs = pf48k
```

Description: Sampling rate selection sent in the channel A transmitted professional channel status.

ProAFsScl

Command Syntax: SR1.DigIO.ProAFsScl

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProAFsScl
SR1.DigIO.ProAFsScl = False
```

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

ProAIncCode

Command Syntax: SR1.DigIO.ProAIncCode

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProAIncCode
SR1.DigIO.ProAIncCode = False
```

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

ProAIncTime

Command Syntax: SR1.DigIO.ProAIncTime

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProAIncTime
SR1.DigIO.ProAIncTime = True
```

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

ProALcl

Command Syntax: SR1.DigIO.ProALcl

Property: Integer

Example:
```
val = SR1.DigIO.ProALcl
SR1.DigIO.ProALcl = 100
```

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

ProALocked

Command Syntax: SR1.DigIO.ProALocked

Property: ProLock enumeration {plNotIndicated=0 | plpUnlocked=1}

Example:
```
val = SR1.DigIO.ProALocked
SR1.DigIO.ProALocked = plNotIndicated
```

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

ProAMCh

Command Syntax: SR1.DigIO.ProAMCh

Property: ProMultiCh enumeration {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}

Example:
```
val = SR1.DigIO.ProAMCh
SR1.DigIO.ProAMCh = pcmMode0
```

Description: Status of Multichannel Mode selection sent with channel A transmitted professional channel status.

ProARef

Command Syntax: SR1.DigIO.ProARef

Property: ProRef enumeration {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}

Example:
```
val = SR1.DigIO.ProARef
SR1.DigIO.ProARef = prGrade1
```

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

ProASrc

Command Syntax: SR1.DigIO.ProASrc

Property: String

Example:
```
val = SR1.DigIO.ProASrc
SR1.DigIO.ProASrc = srs2
```

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

ProATime

Command Syntax: SR1.DigIO.ProATime

Property: Integer

Example:
```
val = SR1.DigIO.ProATime
SR1.DigIO.ProATime = 100
```

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

ProAUser

Command Syntax: SR1.DigIO.ProAUser

Property: ProUser enumeration {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4 | puRsvdForMetadata=5}

Example:
```
val = SR1.DigIO.ProAUser
SR1.DigIO.ProAUser = puNoUserInfo
```

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

ProAWdLen

Command Syntax: SR1.DigIO.ProAWdLen

Property: ProWdLen enumeration {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}

Example:
```
val = SR1.DigIO.ProAWdLen
SR1.DigIO.ProAWdLen = pwl24
```

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

ProBAux

Command Syntax: SR1.DigIO.ProBAux

Property: ProAux enumeration {pa20bitNotDef=0 | pa24bitMainAud=1 | pa20bitCoordSig=2 | paReserved=3}

Example:
```
val = SR1.DigIO.ProBAux
SR1.DigIO.ProBAux = pa20bitNotDef
```

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

ProBCh

Command Syntax: SR1.DigIO.ProBCh

Property: Integer

Example:
```
val = SR1.DigIO.ProBCh
SR1.DigIO.ProBCh = 1
```

Description Channel # setting of the channel B transmitted professional channel status.

ProBChMode

Command Syntax: SR1.DigIO.ProBChMode

Property: ProChMode enumeration {pcmNotIndicated=0 | pcm2Ch=1 | pcmMono=2 | pcmPriSec=3 | pcmStereo=4 | pcmRsvd1=5 | pcmRsvd2=6 | pcm2FsMono=7 | pcm2FsLeft=8 | pcm2FsRight=9 | pcmMultiCh=10}

Example:
```
val = SR1.DigIO.ProBChMode
SR1.DigIO.ProBChMode = pcmMono
```

Description: Channel mode for the channel B transmitted professional channel status.

ProBConf0

Command Syntax: SR1.DigIO.ProBConf0

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProBConf0
SR1.DigIO.ProBConf0 = True
```

Description: Status of the confidence (bytes 0-5) flag for the channel B transmitted professional channel status.

ProBConf1

Command Syntax: SR1.DigIO.ProBConf1

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProBConf1
SR1.DigIO.ProBConf1 = False
```

Description: Status of the confidence (bytes 6-13) flag for the channel B transmitted professional channel status.

ProBConf2

Command Syntax: SR1.DigIO.ProBConf2

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProBConf2
SR1.DigIO.ProBConf2 = True
```

Description: Status of the confidence (bytes 14-17) flag for the channel B transmitted professional channel status.

ProBConf3

Command Syntax: SR1.DigIO.ProBConf3

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ProBConf3
SR1.DigIO.ProBConf3 = False
```

Description: Status of the confidence (bytes 18-21) flag for the channel B transmitted professional channel status.

ProBCrc

Command Syntax: SR1.DigIO.ProBCrc

Property: ProCRC enumeration {pcStatic=0 | pcCorrect=1 | pcIncorrect=2 | pcZero=3}

Example:

```
val = SR1.DigIO.ProBCrc
SR1.DigIO.ProBCrc = pcCorrect
```

Description: CRC sending mode for the channel B transmitted professional channel status.

ProBData

Command Syntax: SR1.DigIO.ProBData

Property: DigitalData enumeration {ddPCM=0 | ddData=1}

Example:

```
val = SR1.DigIO.ProBData
SR1.DigIO.ProBData = ddPCM
```

Description: PCM Audio/Data status for the channel B transmitted professional channel status.

ProBDest

Command Syntax: SR1.DigIO.ProBDest

Property: String

Example:

```
val = SR1.DigIO.ProBDest
SR1.DigIO.ProBDest = srs1
```

Description: 4 character destination label for the the channel B transmitted professional channel status.

ProBEmph

Command Syntax: SR1.DigIO.ProBEmph

Property: ProEmph enumeration {peNotIndicated=0 | peNone=1 | pe5015=2 | peCCITT=3}

Example:

```
val = SR1.DigIO.ProBEmph
SR1.DigIO.ProBEmph = peNone
```

Description: Emphasis mode selection for the channel B transmitted professional channel status.

ProBFs

Command Syntax: SR1.DigIO.ProBFs

Property: ProFs enumeration {pfNotIndicated=0 | pf48k=1 | pf44k=2 | pf32k=3 | pf24k=4 | pf96k=5 | pf192k=6 | pf22k=7 | pf88k=8 | pf176k=9}

Example:

```
val = SR1.DigIO.ProBFs
SR1.DigIO.ProBFs = pf48k
```

Description: Sampling rate selection sent in the channel B transmitted professional channel status.

ProBFsScl

Command Syntax: SR1.DigIO.ProBFsScl

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProBFsScl
SR1.DigIO.ProBFsScl = False
```

Description: Status of the "/1.001" flag sent with the channel B transmitted professional channel status.

ProBIncCode

Command Syntax: SR1.DigIO.ProBIncCode

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProBIncCode
SR1.DigIO.ProBIncCode = False
```

Description: Status of the "Increment Local Address Code" selection for the channel B transmitted professional channel status.

ProBIncTime

Command Syntax: SR1.DigIO.ProBIncTime

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ProBIncTime
SR1.DigIO.ProBIncTime = True
```

Description: Status of the "Increment Time Code" selection for the channel B transmitted professional channel status.

ProBLcl

Command Syntax: SR1.DigIO.ProBLcl

Property: Integer

Example:

```
val = SR1.DigIO.ProBLcl
SR1.DigIO.ProBLcl = 100
```

Description: Value of the "Local Address Code" sent with the channel B transmitted professional channel status.

ProBLocked

Command Syntax: SR1.DigIO.ProBLocked

Property: ProLock enumeration {plNotIndicated=0 | plpUnlocked=1}

Example:

```
val = SR1.DigIO.ProBLocked
SR1.DigIO.ProBLocked = plNotIndicated
```

Description: Status of the "Locked" selection for the channel B transmitted professional channel status.

ProBMCh

Command Syntax: SR1.DigIO.ProBMCh

Property: ProMultiCh enumeration {pmcUndef=0 | pmcMode0=1 | pmcMode1=2 | pmcMode2=3 | pmcMode3=4 | pmcUserDef=5}

Example:

```
val = SR1.DigIO.ProBMCh
SR1.DigIO.ProBMCh = pcmMode0
```

Description: Status of Multichannel Mode selection sent with channel B transmitted professional channel status.

ProBRef

Command Syntax: SR1.DigIO.ProBRef

Property: ProRef enumeration {prNotRef=0 | prGrade1=1 | prGrade2=2 | prRsvd=3}

Example:

```
val = SR1.DigIO.ProBRef
SR1.DigIO.ProBRef = prGrade1
```

Description: Reference Signal selection of the channel B transmitted professional channel status.

ProBSrc

Command Syntax: SR1.DigIO.ProBSrc

Property: String

Example:

```
val = SR1.DigIO.ProBSrc
SR1.DigIO.ProBSrc = srs2
```

Description: 4 character Source Label sent with the channel B transmitted professional channel status.

ProBTime

Command Syntax: SR1.DigIO.ProBTime

Property: Integer

Example:

```
val = SR1.DigIO.ProBTime
SR1.DigIO.ProBTime = 100
```

Description: Time Code sent with the channel B transmitted professional channel status.

ProBUser

Command Syntax: SR1.DigIO.ProBUser

Property: ProUser enumeration {puNoUserInfo=0 | pu192bits=1 | puAES18=2 | puUserDef=3 | puIEC=4 | puRsvdForMetadata=5}

Example:

```
val = SR1.DigIO.ProBUser
SR1.DigIO.ProBUser = puNoUserInfo
```

Description: User Bits mode sent with the channel B transmitted professional channel status.

ProBWdLen

Command Syntax: SR1.DigIO.ProBWdLen

Property: ProWdLen enumeration {pwlNotIndcated=0 | pwl24=1 | pwl23=2 | pwl22=3 | pwl21=4 | pwl20=5}

Example:

```
val = SR1.DigIO.ProBWdLen
SR1.DigIO.ProBWdLen = pwl24
```

Description: Word Length indication sent with the channel B transmitted professional channel status.

Channel Status Commands (Consumer)

ConACat

Command Syntax: SR1.DigIO.ConACat

Property: ConCategory enumeration {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:

```
val = SR1.DigIO.ConACat
SR1.DigIO.ConACat = catLaser
```

Description: Consumer category code indicated by the Channel A digital audio received status bits.

.

ConACh

Command Syntax: SR1.DigIO.ConACh

Property: Integer

Example:

```
val = SR1.DigIO.ConACh
SR1.DigIO.ConACh = 1
```

Description: Channel A consumer Channel #. (0 = N/A, 1=A, 2=B, etc.)

ConAClkAcc

Command Syntax: SR1.DigIO.ConAClkAcc

Property: ConClockAcc enumeration {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateneFs=3}

Example:

```
val = SR1.DigIO.ConAClkAcc
SR1.DigIO.ConAClkAcc = ccaLevel2
```

Description: Channel A received consumer Clock Accuracy field

ConACopyright

Command Syntax: SR1.DigIO.ConACopyright

Property: ConCopyright enumeration {ccCopyright=0 | ccNoCopyright=1}

Example:

```
val = SR1.DigIO.ConACopyright
SR1.DigIO.ConACopyright = ccNoCopyright
```

Description: Channel A consumer received Copyright Bit.

ConAData

Command Syntax: SR1.DigIO.ConAData

Property: DigitalData enumeration {ddPCM=0 | ddData=1}

Example:

```
val = SR1.DigIO.ConAData
SR1.DigIO.ConAData = ddPCM
```

Description: Channel A consumer received Audio Sample field.

ConAEmph

Command Syntax: SR1.DigIO.ConAEmph

Property: ConEmph enumeration {ceNone=0 | ce5015=1}

Example:

```
val = SR1.DigIO.ConAEmph
SR1.DigIO.ConAEmph = ceNone
```

Description: Channel A consumer received Emphasis field.

ConAFs

Command Syntax: SR1.DigIO.ConAFs

Property: ConFs enumeration {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}

Example:

```
val = SR1.DigIO.ConAFs
SR1.DigIO.ConAFs = cf48k
```

Description: Channel A consumer received Sampling Frequency field.

ConAFsOrig

Command Syntax: SR1.DigIO.ConAFsOrig

Property: ConOrigFs enumeration {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:

```
val = SR1.DigIO.ConAFsOrig
SR1.DigIO.ConAFsOrig = cof44k
```

Description: Channel A consumer received "Original Fs" field.

ConAPreRec

Command Syntax: SR1.DigIO.ConAPreRec

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ConAPreRec
SR1.DigIO.ConAPreRec = True
```

Description: Channel A consumer received "Pre-Recorded" status.

ConASrc

Command Syntax: SR1.DigIO.ConASrc

Property: Integer

Example:

```
val = SR1.DigIO.ConASrc
SR1.DigIO.ConASrc = 3
```

Description: Channel A consumer received "Source #" field.

ConAWdLen

Command Syntax: SR1.DigIO.ConAWdLen

Property: ConWordLen enumeration {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 | cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}

Example:

```
val = SR1.DigIO.ConAWdLen
SR1.DigIO.ConAWdLen = cwa24
```

Description: Channel A consumer received Word Length field.

ConBCh

Command Syntax: SR1.DigIO.ConBCh

Property: Integer

Example:

```
val = SR1.DigIO.ConBCh
SR1.DigIO.ConBCh = 1
```

Description: Channel B consumer Channel #. (0 = N/A, 1=A, 2=B, etc.)

ConBClkAcc

Command Syntax: SR1.DigIO.ConBClkAcc

Property: ConClockAcc enumeration {ccaLevel2=0 | ccaLevel1=1 | ccaLevel3=2 | ccaFrameRateneFs=3}

Example:

```
val = SR1.DigIO.ConBClkAcc
SR1.DigIO.ConBClkAcc = ccaLevel2
```

Description: Channel B received consumer Clock Accuracy field

ConBCopyright

Command Syntax: SR1.DigIO.ConBCopyright

Property: ConCopyright enumeration {ccCopyright=0 | ccNoCopyright=1}

Example:

```
val = SR1.DigIO.ConBCopyright
SR1.DigIO.ConBCopyright = ccNoCopyright
```

Description: Channel B consumer received Copyright Bit.

ConBData

Command Syntax: SR1.DigIO.ConBData

Property: DigitalData enumeration {ddPCM=0 | ddData=1}

Example:

```
val = SR1.DigIO.ConBData
SR1.DigIO.ConBData = ddPCM
```

Description: Channel B consumer received Audio Sample field.

ConBEmph

Command Syntax: SR1.DigIO.ConBEmph

Property: ConEmph enumeration {ceNone=0 | ce5015=1}

Example:
```
val = SR1.DigIO.ConBEmph
SR1.DigIO.ConBEmph = ceNone
```

Description: Channel B consumer received Emphasis field.

ConBFs

Command Syntax: SR1.DigIO.ConBFs

Property: ConFs enumeration {cf22k=0 | cf44k=1 | cf88k=2 | cf176k=3 | cf24k=4 | cf48k=5 | cf96k=6 | cf192k=7 | cf32k=8 | cfNotIndicated=9}

Example:
```
val = SR1.DigIO.ConBFs
SR1.DigIO.ConBFs = cf48k
```

Description: Channel B consumer received Sampling Frequency field.

ConBFsOrig

Command Syntax: SR1.DigIO.ConBFsOrig

Property: ConOrigFs enumeration {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:
```
val = SR1.DigIO.ConBFsOrig
SR1.DigIO.ConBFsOrig = cof44k
```

Description: Channel B consumer received "Original Fs" field.

ConBPreRec

Command Syntax: SR1.DigIO.ConBPreRec

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ConBPreRec
SR1.DigIO.ConBPreRec = True
```

Description: Channel B consumer received "Pre-Recorded" status.

ConBSrc

Command Syntax: SR1.DigIO.ConBSrc

Property: Integer

Example:
```
val = SR1.DigIO.ConBSrc
SR1.DigIO.ConBSrc = 3
```

Description: Channel B consumer received "Source #" field.

ConBWdLen

Command Syntax: SR1.DigIO.ConBWdLen

Property: ConWordLen enumeration {cwlNotIndicated=0 | cwa16=1 | cwa17=2 | cwa18=3 | cwa19=4 | cwa20=5 | cwa21=6 | cwa22=7 | cwa23=8 | cwa24=9}

Example:

```
val = SR1.DigIO.ConBWdLen
SR1.DigIO.ConBWdLen = cwa24
```

Description: Channel B consumer received Word Length field.

Impairment Commands

ClockOutJitter

Command Syntax: SR1.DigIO.ClockOutJitter

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ClockOutJitter
SR1.DigIO.ClockOutJitter = False
```

Description: Value of the Rear Panel Clock Jitter Enable.

ImpairCableSim

Command Syntax: SR1.DigIO.ImpairCableSim

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ImpairCableSim
SR1.DigIO.ImpairCableSim = False
```

Description: Status of the Cable Simulator on the Output Impairment tab of the Digital I/O panel.

ImpairCarrierInvert

Command Syntax: SR1.DigIO.ImpairCarrierInvert

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ImpairCarrierInvert
SR1.DigIO.ImpairCarrierInvert = False
```

Description: Carrier invert status on the Impairment tab of the Digital I/O panel.

ImpairCommonMode

Command Syntax: SR1.DigIO.ImpairCommonMode

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigIO.ImpairCommonMode
SR1.DigIO.ImpairCommonMode = False
```

Description: On/Off status of the Common Mode Sine on the Output Impairment tab of the Digitals I/O panel.

ImpairCommonModeAmp

Command Syntax: SR1.DigIO.ImpairCommonModeAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigIO.ImpairCommonModeAmp("")
SR1.DigIO.ImpairCommonModeAmp("") = 1.1
```

Description: Amplitude of the Common Mode Sine impairment signal.

ImpairCommonModeFreq

Command Syntax: SR1.DigIO.ImpairCommonModeFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigIO.ImpairCommonModeFreq("")
SR1.DigIO.ImpairCommonModeFreq("") = 1200
```

Description: Frequency of the Common Mode Sine impairment signal.

ImpairJitterAmp

Command Syntax: SR1.DigIO.ImpairJitterAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigIO.ImpairJitterAmp("")
SR1.DigIO.ImpairJitterAmp("") = 0.012
```

Description: Amplitude of the digital audio output impairment jitter signal.

ImpairJitterFreq

Command Syntax: SR1.DigIO.ImpairJitterFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigIO.ImpairJitterFreq("")
SR1.DigIO.ImpairJitterFreq("") = 1000
```

Description: Frequency of the digital audio output impairment jitter signal.

ImpairJitterWaveform

Command Syntax: SR1.DigIO.ImpairJitterWaveform

Property: JitterType enumeration {jtOff=0 | jtSine=1 | jtSquare=2 | jtNoise=3 | jtNoiseBP=4}

Example:

```
val = SR1.DigIO.ImpairJitterWaveform
SR1.DigIO.ImpairJitterWaveform = jtNoise
```

Description: Waveform type of the digital audio output jitter impairment signal.

SetJitterEq

Command Syntax: SR1.DigIO.SetJitterEq(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.DigIO.SetJitterEq("FileName")
```

Description Loads an EQ file for jitter output (does not apply to uniform noise). Use "" as filename to remove EQ.

ImpairNormalMode

Command Syntax: SR1.DigIO.ImpairNormalMode

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ImpairNormalMode
SR1.DigIO.ImpairNormalMode = False
```

Description: On/off status of the digital audio output Normal Mode Noise impairment signal.

ImpairNormalModeAmp

Command Syntax: SR1.DigIO.ImpairNormalModeAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigIO.ImpairNormalModeAmp("")
SR1.DigIO.ImpairNormalModeAmp("") = 1.2
```

Description: Amplitude of the digital audio output Normal Mode Noise impairment signal.

ImpairRiseFallTime

Command Syntax: SR1.DigIO.ImpairRiseFallTime(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Example:

```
val = SR1.DigIO.ImpairRiseFallTime("")
SR1.DigIO.ImpairRiseFallTime("") = Value
```

Description: Rise and fall time of digital carrier.

ImpairSendInvalidA

Command Syntax: SR1.DigIO.ImpairSendInvalidA

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ImpairSendInvalidA
SR1.DigIO.ImpairSendInvalidA = Value
```

Description: `Controls if invalid status is sent on channel A.`

ImpairSendInvalidB

Command Syntax: SR1.DigIO.ImpairSendInvalidB

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigIO.ImpairSendInvalidB
SR1.DigIO.ImpairSendInvalidB = Value
```

Description: `Controls if invalid status is sent on channel B.`

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

OpenChannelStatusBitsForm

Command Syntax: SR1.DigIO.OpenChannelStatusBitsForm()

Command Argument(s): None

Example: `Call SR1.DigIO.OpenChannelStatusBitsForm()`

Description: Opens a channel status bits form on the current page of the page control.

OpenChannelStatusBitsFormwID

Command Syntax: SR1.DigIO.OpenChannelStatusBitsFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.DigIO.OpenChannelStatusBitsFormwID()`

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

CloseChannelStatusBitsForm

Command Syntax: SR1.DigIO.CloseChannelStatusBitsForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.DigIO.CloseChannelStatusBitsForm(10)`

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

CloseChannelStatusBitsForms

Command Syntax: SR1.DigIO.CloseChannelStatusBitsForms()

Command Argument(s): None

Example: `Call SR1.DigIO.CloseChannelStatusBitsForms()`

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

ChannelStatusBitsFormCount

Command Syntax: SR1.DigIO.ChannelStatusBitsFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.DigIO.ChannelStatusBitsFormCount()`

Description: Returns the number of open channel Status forms on all pages of the page control

ChannelStatusBitsFormID

Command Syntax: SR1.DigIO.ChannelStatusBitsFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.DigIO.ChannelStatusBitsFormID(0)`

Description: Returns the FormID of the Indexth channel status form. Index = 0 corresponds to the first form.

OpenUserStatusBitsForm

Command Syntax: SR1.DigIO.OpenUserStatusBitsForm()

Command Argument(s): None

Example: `Call SR1.DigIO.OpenUserStatusBitsForm()`

Description: Opens a User status bits form on the current page of the page control.

OpenUserStatusBitsFormwID

Command Syntax: SR1.DigIO.OpenUserStatusBitsFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.DigIO.OpenUserStatusBitsFormwID()`

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

CloseUserStatusBitsForm

Command Syntax: SR1.DigIO.CloseUserStatusBitsForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.DigIO.CloseUserStatusBitsForm(10)`

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

CloseUserStatusBitsForms

Command Syntax: SR1.DigIO.CloseUserStatusBitsForms()

Command Argument(s): None

Example: `Call SR1.DigIO.CloseUserStatusBitsForms()`

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

UserStatusBitsFormCount

Command Syntax: SR1.DigIO.UserStatusBitsFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.DigIO.UserStatusBitsFormCount()`

Description: Returns the number of open User Status forms on all pages of the page control

UserStatusBitsFormID

Command Syntax: SR1.DigIO.UserStatusBitsFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.DigIO.UserStatusBitsFormID(0)`

Description: Returns the FormID of the Indexth Use.r status form. Index = 0 corresponds to the first form.

2.4.5 Sweep

Object:	**SR1.Sweep**
Object Argument(s):	None
Description:	Commands related to the Sweep Configuration Panel.

General Sweep Commands

Start

Command Syntax: SR1.Sweep.Start()

Example: `Call SR1.Sweep.Start()`

Description: Starts the currently configured sweep.

FreeRun

Command Syntax: SR1.Sweep.FreeRun()

Example: `Call SR1.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: SR1.Sweep.Pause()

Example: `Call SR1.Sweep.Pause()`

Description: Pauses either the current sweep or free-run, depending on the current state.

pauseResume

Command Syntax: SR1.Sweep.pauseResume()

Example: `Call SR1.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: SR1.Sweep.Resume()

Example: `Call SR1.Sweep.Resume()`

Description: Resumes either a sweep or free-run that is currently paused.

GetState

Command Syntax: SR1.Sweep.GetState()

Return Value: *State* as SweepState enumeration {ssFreeRunActive=0 | ssFreeRunPaused=1 | ssSweepActive=2 | ssSweepPaused=3 | ssSweepFinished=4}

Example: `State = SR1.Sweep.GetState()`

Description: Current state of the Sweep Controller.

PreSweepDelay

Command Syntax: SR1.Sweep.PreSweepDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Sweep.PreSweepDelay("")
SR1.Sweep.PreSweepDelay("") = 2.2
```

Description: Value of the Pre-Sweep Delay.

Repeat

Command Syntax: SR1.Sweep.Repeat

Property: Integer

Example:
```
val = SR1.Sweep.Repeat
SR1.Sweep.Repeat = False
```

Description: On/off status of the Sweep Repeat.

Timeout

Command Syntax: SR1.Sweep.Timeout(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Sweep.Timeout("s")
SR1.Sweep.Timeout("s") = 10
```

Description: Sweep Timeout value.

Sweep Source Commands

Sources

Command Syntax: SR1.Sweep.Sources

Property: Integer (reasd-only)

Example:
```
val = SR1.Sweep.Sources
```

Description: Returns 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: SR1.Sweep.SourceCfgExternal(*Axis*, *MeasID*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}
MeasID as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `Call SR1.Sweep.SourceCfgExternal(swpInner,` msAnlgFreqA)

Description: Configures the specified sweep axis (inner or outer) as an external based on the measurement referenced by the *MeasID* argument. Currently, only the inner axis supports external sweeps.

SourceCfgExternalRdg

Command Syntax: SR1.Sweep.SourceCfgExternalRdg(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Return Value: *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.Sweep.SourceCfgExternalRdg(swpInner)`

Description: 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: SR1.Sweep.SourceCfgInternal(*Axis*, *ParamID*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}
ParamID as String

Example: `Call SR1.Sweep.SourceCfgInternal(swpInner, "SR1.AnlgGen.`

Description: Configures the specified sweep axis as an internal sweep based on the parameter specified by the string *ParamID*.

SourceCfgInternalRdg

Command Syntax: SR1.Sweep.SourceCfgInternalRdg(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Return Value: *ParamID* as String

Example: `ParamID = SR1.Sweep.SourceCfgInternalRdg(swpInner)`

Description: Returns 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: SR1.Sweep.SourceCfgNone(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Example: `Call SR1.Sweep.SourceCfgNone(swpInner)`

Description: Removes any sweep source from the specified sweep axis.

SourceCfgRdg

Command Syntax: SR1.Sweep.SourceCfgRdg(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Return Value: *Type* as eSweepType enumeration {estNone=0 | estInternal=1 | estExternal=2 | estTime=3 | estSwitcher=4}

Example: `Type = SR1.Sweep.SourceCfgRdg(swpInner)`

Description: Returns the type of sweep source connected to the specified axis.

SourceCfgSwitcher

Command Syntax: SR1.Sweep.SourceCfgSwitcher(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Example: `Call SR1.Sweep.SourceCfgSwitcher(swpOuter)`

Description: Sets the sweep source for the specified axis to a Switcher Sweep. Currently, only the outer axis supports switcher sweeps.

SourceCfgTime

Command Syntax: SR1.Sweep.SourceCfgTime(*Axis*, *TimeID*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}
TimeID as TimeType enumeration {ttIntersampleDelay=0 | ttCorrectedTime=1}

Example: `Call SR1.Sweep.SourceCfgTime(swpInner, ttIntersampleDelay)`

Description: Sets the sweep source for the specified axis to a Time Sweep using the specified time mode. Currently, only the inner axis supports time sweeps.

SourceCfgTimeRdg

Command Syntax: SR1.Sweep.SourceCfgTimeRdg(*Axis*)

Command Argument(s): *Axis* as SweepAxis enumeration {swpInner=0 | swpOuter=1}

Return Value: *TimeID* as TimeType enumeration {ttIntersampleDelay=0 | ttCorrectedTime=1}

Example: `TimeID = SR1.Sweep.SourceCfgTimeRdg(swpInner)`

Description: `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: SR1.Sweep.DataCfg(*DataID*, *MeasID*)

Command Argument(s): *DataID* as Integer
MeasID as MeasID enumeration{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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `Call SR1.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: SR1.Sweep.DataCfgRdg(*DataID*)

Command Argument(s): *DataID* as Integer

Return Value: *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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:
```
MeasID = SR1.Sweep.DataCfgRdg(0)
```

Description: Returns 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.

External Sweep Commands

StartMethod

Command Syntax: SR1.Sweep.StartMethod

Property: StartMethods enumeration {ssStartWithinTol=0 | ssStartWithinRange=1 | ssStartAnywhere=2}

Example:
```
val = SR1.Sweep.StartMethod
SR1.Sweep.StartMethod = ssStartWithinRange
```

Description: External sweep "Start Sweep At" selection.

StopMethod

Command Syntax: SR1.Sweep.StopMethod

Property: StopMethods enumeration {ssStopWithinTol=0 | ssStopOnReturn=1}

Example:

```
val = SR1.Sweep.StopMethod
SR1.Sweep.StopMethod = ssStopOnReturn
```

Description: External sweep "Stop Sweep At" selection.

ExtMinLevelCfg

Command Syntax: SR1.Sweep.ExtMinLevelCfg(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `Call SR1.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: SR1.Sweep.ExtMinLevelCfgRdg()

Return Value: *MeasID* as MeasID enumeration {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 | 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 |
msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.Sweep.ExtMinLevelCfgRdg()`

Description 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

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

SR1.Sweep.OpenSettlingForm

Command Syntax: SR1.Sweep.OpenSettlingForm()

Command Argument(s): None

Example: `Call SR1.Sweep.OpenSettlingForm()`

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

SR1.Sweep.OpenSettlingFormwID

Command Syntax: SR1.Sweep.OpenSettlingFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Sweep.OpenSettlingFormwID()`

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

SR1.Sweep.CloseSettlingForm

Command Syntax: SR1.Sweep.CloseSettlingForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.Sweep.CloseSettlingForm(10)`

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

SR1.Sweep.CloseSettlingForms

Command Syntax: SR1.Sweep.CloseSettlingForms()

Command Argument(s): None

Example: `Call SR1.Sweep.CloseSettlingForms()`

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

SR1.Sweep.SettlingFormCount

Command Syntax: SR1.Sweep.SettlingFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Sweep.SettlingFormCount()`

Description: Returns the number of open settling forms on all pages of the page control

SR1.Sweep.SettlingFormID

Command Syntax: SR1.Sweep.SettlingFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.Sweep.SettlingFormID(0)`

Description: Returns the FormID of the Indexth settling form. Index = 0 corresponds to the first form.

2.4.5.1 Sweep Source

Object:	**SR1.Sweep.Source(*Axis*)**
Object Argument(s):	*Axis* as Integer {swpInner=0 | swpOuter=1}
Description:	Commands related to the configuration of the inner or outer sweep axis.

Internal Sweep Source Commands

InternLinStep

Command Syntax: SR1.Sweep.Source(i).InternLinStep(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Sweep.Source(0).InternLinStep("Hz")
SR1.Sweep.Source(0).InternLinStep("Hz") = 100
```

Description: Value of linear sweep step for the internal sweep source associated with the specified axis.

InternLogStep

Command Syntax: SR1.Sweep.Source(i).InternLogStep(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Sweep.Source(1).InternLogStep("")
SR1.Sweep.Source(1).InternLogStep("") = 1.1
```

Description: Value of the log sweep step for the internal sweep source associated with the specified axis.

IntNumSteps

Command Syntax: SR1.Sweep.Source(i).IntNumSteps

Property: Integer

Example:
```
val = SR1.Sweep.Source(0).IntNumSteps
SR1.Sweep.Source(0).IntNumSteps = 100
```

Description: Number of steps of internal sweep source associated with the specified axis.

Log

Command Syntax: SR1.Sweep.Source(i).Log

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Sweep.Source(0).Log
SR1.Sweep.Source(0).Log = True
```

Description: Log sweep mode of internal sweep source associated with the specified axis.

TableName

Command Syntax: SR1.Sweep.Source(i).TableName

Property: String

Example:

```
val = SR1.Sweep.Source(1).TableName
SR1.Sweep.Source(1).TableName = "MyTable.txt"
```

Description: Table filename used in conjunction with table-driven internal sweeps.

TableColIndex

Command Syntax: SR1.Sweep.Source(i).TableColIndex

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).TableColIndex
SR1.Sweep.Source(1).TableColIndex = 2
```

Description: Index of the column that will be used to obtain table-sweep data. The first column has index 1, the second, 2, etc.

LoadSweepTable

Command Syntax: SR1.Sweep.Source(i).LoadSweepTable(*FileName*, *ColIndex*)

Command Argument(s): *FileName* as String
ColIndex as Integer

Example:

```
Call SR1.Sweep.Source(0).LoadSweepTable("MyTable.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: SR1.Sweep.Source(i).ExternStep(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).ExternStep("FFS")
SR1.Sweep.Source(0).ExternStep("FFS") = 0.01
```

Description: Value of the external sweep step for the inner sweep axis.

ExternStepRel

Command Syntax: SR1.Sweep.Source(i).ExternStepRel(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Example:

```
val = SR1.Sweep.Source(0).ExternStepRel("%")
SR1.Sweep.Source(0).ExternStepRel("%") = 4
```

Description: Value of the external relative sweep step for the inner sweep axis.

UseExternStepRel

Command Syntax: SR1.Sweep.Source(i).UseExternStepRel

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Sweep.Source(i).UseExternStepRel
SR1.Sweep.Source(0).UseExternStepRel = True
```

Description: Determines whether the external sweep will used the relative step-size (true) or the absolute step size (false).

MinLevel

Command Syntax: SR1.Sweep.Source(i).MinLevel(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).MinLevel("Vrms")
SR1.Sweep.Source(0).MinLevel("Vrms") = 0.001
```

Description: 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: SR1.Sweep.Source(i).StartTol(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).StartTol("Hz")
SR1.Sweep.Source(0).StartTol("Hz") = 10
```

Description: 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: SR1.Sweep.Source(i).StartTolRel(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).StartTolRel("%")
SR1.Sweep.Source(0).StartTolRel("%") = 10
```

Description: 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: SR1.Sweep.Source(i).UseStartTolRel

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Sweep.Source(0).UseStartTolRel
SR1.Sweep.Source(0).UseStartTolRel = True
```

Description: Determines whether the relative (true) or absolute (false) start tolerance is used when deciding to start an external sweep.

StartVal

Command Syntax: SR1.Sweep.Source(i).StartVal(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).StartVal("Hz")
SR1.Sweep.Source(0).StartVal("Hz") = 20
```

Description: Desired starting value for th the external sweep.

StopTol

Command Syntax: SR1.Sweep.Source(i).StopTol(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).StopTol("Hz")
SR1.Sweep.Source(0).StopTol("Hz") = 100
```

Description: 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: SR1.Sweep.Source(i).StopTolRel(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Source(0).StopTolRel("pct")
SR1.Sweep.Source(0).StopTolRel("pct") = 10
```

Description: 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: SR1.Sweep.Source(i).UseStopTolRel

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Sweep.Source(0).UseStopTolRel
SR1.Sweep.Source(0).UseStopTolRel = False
```

Description: Determines whether the relative (true) or absolute (false) stop tolerance is used when deciding to stop an external sweep.

StopVal

Command Syntax: SR1.Sweep.Source(i).StopVal(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

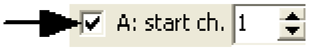
Example:

```
val = SR1.Sweep.Source(0).StopVal("Vrms")
SR1.Sweep.Source(0).StopVal("Vrms") = 5.0
```

Description: 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, switcher sweeps are only allowed on the outer sweep axis.



☑ A: start ch. 1

InSwitchEnableChA

Command Syntax: SR1.Sweep.Source(i).InSwitchEnableChA

Property: Boolean {False=0 | True=1}

Example:

```
val = SR1.Sweep.Source(1).InSwitchEnableChA
SR1.Sweep.Source(1).InSwitchEnableChA = True
```

Description: Enable status of the Input Switch channel A sweep.

InSwitchEnableChB

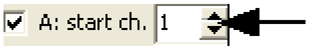
Command Syntax: SR1.Sweep.Source(i).InSwitchEnableChB

Property: Boolean {False=0 | True=1}

Example:

```
val = SR1.Sweep.Source(1).InSwitchEnableChB
SR1.Sweep.Source(1).InSwitchEnableChB = False
```

Description: Enable status of the Input Switch channel B sweep.



☑ A: start ch. 1

InSwitchStartChA

Command Syntax: SR1.Sweep.Source(i).InSwitchStartChA

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).InSwitchStartChA
SR1.Sweep.Source(1).InSwitchStartChA = 6
```

Description: Starting logical channel number for the channel A input switcher sweep.

InSwitchStartChB

Command Syntax: SR1.Sweep.Source(i).InSwitchStartChB

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).InSwitchStartChB
SR1.Sweep.Source(1).InSwitchStartChB = 33
```

Deescription: Starting logical channel number for the channel B input switcher sweep.

InSwitchName

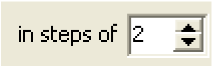
Command Syntax: SR1.Sweep.Source(i).InSwitchName

Property: String

Example:

```
val = SR1.Sweep.Source(1).InSwitchName
SR1.Sweep.Source(1).InSwitchName = "BNC"
```

Description: Network name for the input sweep



in steps of 2

InSwitchStep

Command Syntax: SR1.Sweep.Source(i).InSwitchStep

Property: Integer

Example:
```
val = SR1.Sweep.Source(1).InSwitchStep
SR1.Sweep.Source(1).InSwitchStep = 4
```

Description: Logical channel number step size for the input switcher sweep.

☐ Bus

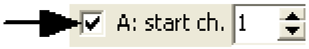
OutSwitchBusChA

Command Syntax: SR1.Sweep.Source(i).OutSwitchBusChA

Property: Boolean {False=0 | True=1}

Example:
```
val = SR1.Sweep.Source(1).OutSwitchBusChA
SR1.Sweep.Source(1).OutSwitchBusChA = True
```

Description: 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.



☑ A: start ch. 1

OutSwitchEnableChA

Command Syntax: SR1.Sweep.Source(i).OutSwitchEnableChA

Property: Boolean {False=0 | True=1}

Example:
```
val = SR1.Sweep.Source(1).OutSwitchEnableChA
SR1.Sweep.Source(1).OutSwitchEnableChA = False
```

Description: Enable status of the Output Switch channel A sweep.

OutSwitchEnableChB

Command Syntax: SR1.Sweep.Source(i).OutSwitchEnableChB

Property: Boolean {False=0 | True=1}

Example:
```
val = SR1.Sweep.Source(1).OutSwitchEnableChB
SR1.Sweep.Source(1).OutSwitchEnableChB = True
```

Description: Enable status of the Output Switch channel B sweep.

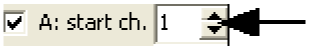
OutSwitchName

Command Syntax: SR1.Sweep.Source(*i*).OutSwitchName

Property: *Value* <string>

Example:
```
val = SR1.Sweep.Source(1).OutSwitchName
SR1.Sweep.Source(1).OutSwitchName = "Switch2"
```

Description: Network name for the output sweep



☑ A: start ch. 1

OutSwitchStartChA

Command Syntax: SR1.Sweep.Source(i).OutSwitchStartChA

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).OutSwitchStartChA
SR1.Sweep.Source(1).OutSwitchStartChA = 1
```

Description: Starting logical channel number for the A-channel output switcher sweep.

OutSwitchStartChB

Command Syntax: SR1.Sweep.Source(i).OutSwitchStartChB

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).OutSwitchStartChB
SR1.Sweep.Source(1).OutSwitchStartChB = 2
```

Description: Starting logical channel number for the B-channel output switcher sweep.

Steps 1

OutSwitchStep

Command Syntax: SR1.Sweep.Source(i).OutSwitchStep

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).OutSwitchStep
SR1.Sweep.Source(1).OutSwitchStep = 2
```

Description: Logical channel number step stize for the output switcher sweep.

Steps 1

SwitchNumSteps

Command Syntax: SR1.Sweep.Source(i).SwitchNumSteps

Property: Integer

Example:

```
val = SR1.Sweep.Source(1).SwitchNumSteps
SR1.Sweep.Source(1).SwitchNumSteps = 10
```

Description: Number of steps in the switcher sweep.

2.4.5.2 Sweep Settling

Object:	**SR1.Sweep.Settling(*i*)**
Object Argument(s):	i as Integer {0 | 1 | 2 | 3 | ... | 116 | 117}
Description:	Commands Related to the settling parameters for the *i*th settler.

The commands below set the properties for the settler SR1.Sweep.Settling(i). To find the index i corresponding to a given measurement use the command:

```
SR1.Instrument.SettlerIndex(MeasId)
```

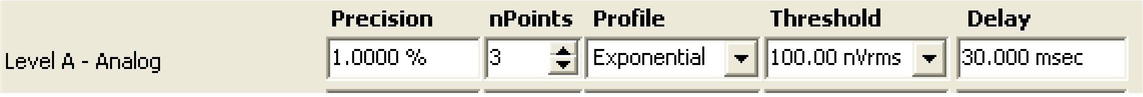
See the Instrument section for a list of measurement IDs. For instance, to get the settler index corresponding to the FFT spectrum for analyzer A1 send:

```
idx = SR1.Instrument.SettlerIndex(msA1FFTspectrum)
```

the return value in idx is 8.

And the settling parameters for this measurement can then be accessed using:

```
SR1.Sweep.Settling(idx).Command
```



	Precision	nPoints	Profile	Threshold	Delay
Level A - Analog	1.0000 %	3	Exponential	100.00 nVrms	30.000 msec

Delay

Command Syntax: SR1.Sweep.Settling(i).Delay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Settling(27).Delay("s")
SR1.Sweep.Settling(27).Delay("s") = 0.030
```

Description: 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.

Floor

Command Syntax: SR1.Sweep.Settling(i).Floor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Settling(51).Floor("Vrms")
SR1.Sweep.Settling(51).Floor("Vrms") = 0.010
```

Description: 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: SR1.Sweep.Settling(i).Method

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.Sweep.Settling(70).Method
SR1.Sweep.Settling(70).Method = stlNone
```

Description: Algorithm type for the specified settling object.

N

Command Syntax: SR1.Sweep.Settling(i).N

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Sweep.Settling(9).N
SR1.Sweep.Settling(9).N = 5
```

Description: Number of points for the specified settling object.

Tolerance

Command Syntax: SR1.Sweep.Settling(i).Tolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Sweep.Settling(14).Tolerance("%")
SR1.Sweep.Settling(14).Tolerance("%") = 1
```

Description: Tolerance window for the specified settling object.

2.4.6 Analyzer

Object:	**SR1.Alyzr(*i*)**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the specified analyzer, either A0 or A1.

Function and Input Commands

Function

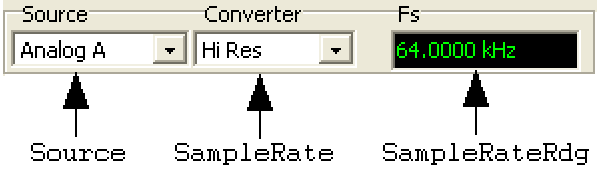
Command Syntax: SR1.Alyzr(i).Function

Command Argument(s):

Property: AnalyzerFunction enumeration {azTimeDomDet=0 | azFFT=1 | azFFT2Ch=2 | azTHD=5 | azIMD=6 | azMultitone=7 | azJitter=9 | azHistogram=10}

Example:
```
val = SR1.Alyzr(0).Function
SR1.Alyzr(0).Function = azTimeDomDet
```

Description: Function (type) of the specified analyzer.



Source: Analog A

Converter: Hi Res

Fs: 64.0000 kHz

SampleRate

Command Syntax: SR1.Alyzr(i).SampleRate

Command Argument(s):

Property: SampleRateEnum enumeration {azHiBW=0 | azHiRes=1 | azISR=2 | azDetOutSR=3}

Example:
```
val = SR1.Alyzr(0).SampleRate
SR1.Alyzr(0).SampleRate = azHiBW
```

Description: Enumerated sample rate of the specified analyzer.

SampleRateRdg

Command Syntax: SR1.Alyzr(i).SampleRateRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Alyzr(0).SampleRateRdg("Hz")
```

Description: Effective sample rate for the given analyzer.

Source

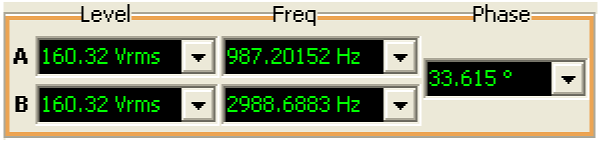
Command Syntax: SR1.Alyzr(i).Source

Command Argument(s):

Property: SignalSource enumeration {azAnlg=0 | azDig=1 | azAnlgA=2 | azAnlgB=3 | azDigA=4 | azDigB=5 | azDetOut=6}

Example:
```
val = SR1.Alyzr(0).Source
SR1.Alyzr(0).Source = azAnlg
```

Description: Analyzer input source selection.



	Level	Freq	Phase
A	160.32 Vrms	987.20152 Hz	33.615 °
B	160.32 Vrms	2988.6883 Hz	

Analyzer Level Commands

Analog

AnlgFreqARdg

Command Syntax: SR1.Alyzr(i).AnlgFreqARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Alyzr(0).AnlgFreqARdg("Hz")
```

Description: Analyzer's analog A-channel frequency reading.

AnlgFreqBRdg

Command Syntax: SR1.Alyzr(i).AnlgFreqBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Alyzr(0).AnlgFreqBRdg("Hz")
```

Description: Analyzer's analog B-channel frequency reading.

AnlgLevelARdg

Command Syntax: SR1.Alyzr(i).AnlgLevelARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).AnlgLevelARdg("Vrms")`
Description: Analyzer's analog A-channel level reading..

AnlgLevelBRdg

Command Syntax: SR1.Alyzr(i).AnlgLevelBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).AnlgLevelBRdg("Vrms")`
Description: Analyzer's analog B-channel level reading..

AnlgPhaseRdg

Command Syntax: SR1.Alyzr(i).AnlgPhaseRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).AnlgPhaseRdg("")`
Description: Analzyer's analog intra-channel phase reading.

Digital

DigFreqARdg

Command Syntax: SR1.Alyzr(i).DigFreqARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).DigFreqARdg("Hz")`
Description: Analyzer's digital A-channel frequency reading.

DigFreqBRdg

Command Syntax: SR1.Alyzr(i).DigFreqBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).DigFreqBRdg("Hz")`
Description: Analyzer's digital B-channel frequency reading.

DigLevelARdg

Command Syntax: SR1.Alyzr(i).DigLevelARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).DigLevelARdg("FFS")`

Description: Analyzer's digital A-channel level reading..

DigLevelBRdg

Command Syntax: SR1.Alyzr(i).DigLevelBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).DigLevelBRdg("")`

Description: Analyzer's digital B-channel level reading..

DigPhaseRdg

Command Syntax: SR1.Alyzr(i).DigPhaseRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

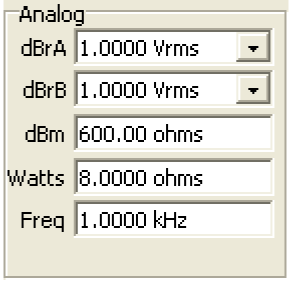
Property: Double (read-only)

Example: `val = SR1.Alyzr(0).DigPhaseRdg("")`

Description: Analzyer's digital audio intra-channel phase reading.

Analyzer Reference Commands

Analog



Analog

dBrA 1.0000 Vrms

dBrB 1.0000 Vrms

dBm 600.00 ohms

Watts 8.0000 ohms

Freq 1.0000 kHz

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 `SR1.Alyzer(0).AnlgdBrA and SR1.Alyzer(1).AnlgdBrA` both change the same quantity.

AnlgdBrA

Command Syntax: SR1.Alyzr(i).AnlgdBrA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).AnlgdBrA("Vrms")
SR1.Alyzr(0).AnlgdBrA("Vrms") = 1.000
```

Description: dBrA reference for the both analyzers.

AnlgdBrB

Command Syntax: SR1.Alyzr(i).AnlgdBrB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).AnlgdBrB("Vrms")
SR1.Alyzr(0).AnlgdBrB("Vrms") = 1.000
```

Description: dBrB reference for the both analyzers.

AnlgFreq

Command Syntax: SR1.Alyzr(i).AnlgFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).AnlgFreq("Hz")
SR1.Alyzr(0).AnlgFreq("Hz") = 1000.0
```

Description: Analog frequency reference for both analyzers.

dBmZ

Command Syntax: SR1.Alyzr(i).dBmZ(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).dBmZ("ohms")
SR1.Alyzr(0).dBmZ("ohms") = 600
```

Description: dBm reference impedance for both analyzers.

WattsZ

Command Syntax: SR1.Alyzr(i).WattsZ(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

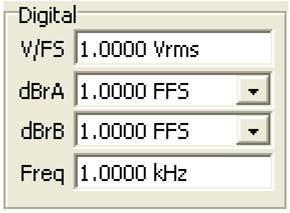
Property: Double

Example:

```
val = SR1.Alyzr(0).WattsZ("")
SR1.Alyzr(0).WattsZ("") = Value
```

Description: 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: SR1.Alyzr(i).DigdBrA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).DigdBrA("Vrms")
SR1.Alyzr(0).DigdBrA("Vrms") = 1.000
```

Description: Digital audio dBrA reference value for both analyzers.

DigdBrB

Command Syntax: SR1.Alyzr(i).DigdBrB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).DigdBrB("FFS")
SR1.Alyzr(0).DigdBrB("FFS") = 1.000
```

Description: Digital audio dBrB reference value for both analyzers.

DigFreq

Command Syntax: SR1.Alyzr(i).DigFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).DigFreq("Hz")
SR1.Alyzr(0).DigFreq("Hz") = 100.0
```

Description: Digital audio frequency reference for both analyzers.

DigVfs

Command Syntax: SR1.Alyzr(i).DigVfs(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).DigVfs("Vrms")
SR1.Alyzr(0).DigVfs("Vrms") = 10
```

Description: Digital audio Volts/Full Scale value for both analyzers.

Form Commands:

The form commands for SR1.Alyzr(i) refer to the currently active sub-analyzer.

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.1 Analyzer Trigger

Object:	**SR1.Alyzr(*i*).Trigger**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the triggering parameters for the specified analyzer.

☑ Enabled

Active

Command Syntax: SR1.Alyzr(i).Trigger.Active

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Trigger.Active
SR1.Alyzr(0).Trigger.Active = False
```

Description: Trigger enabled state for the given analyzer.

Source

Input Channel

Source

Command Syntax: SR1.Alyzr(i).Trigger.Source

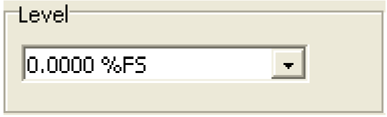
Command Argument(s):

Property: TrgSrc enumeration {trgInputCh=0 | trgOtherCh=1 | trgExt=2 | trgManual=3 | trgGen=4 | trgDigAudBlock=5 | trgBurstA=6 | trgBurstB=7}

Example:

```
val = SR1.Alyzr(0).Trigger.Source
SR1.Alyzr(0).Trigger.Source = trgInputCh
```

Description: Source of the Analyzer's trigger.



Level

0.0000 %FS

AnlgLevelA

Command Syntax: SR1.Alyzr(i).Trigger.AnlgLevelA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Trigger.AnlgLevelA("pctFS")
SR1.Alyzr(0).Trigger.AnlgLevelA("pctFS") = 50.0
```

Description: A-channel analog trigger level.

AnlgLevelB

Command Syntax: SR1.Alyzr(i).Trigger.AnlgLevelB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Trigger.AnlgLevelB("pctFS")
SR1.Alyzr(0).Trigger.AnlgLevelB("pctFS") = 50.0
```

Description: B-channel analog trigger level.

DigLevelA

Command Syntax: SR1.Alyzr(i).Trigger.DigLevelA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Trigger.DigLevelA("pctFS")
SR1.Alyzr(0).Trigger.DigLevelA("pctFS") = 40
```

Description: A-Channel digital audio trigger level.

DigLevelB

Command Syntax: SR1.Alyzr(i).Trigger.DigLevelB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Trigger.DigLevelB("pctFS")
SR1.Alyzr(0).Trigger.DigLevelB("pctFS") = 40
```

Description: B-Channel digital audio trigger level.

JitterLevel

Command Syntax: SR1.Alyzr(i).Trigger.JitterLevel(*ValueUnit*)

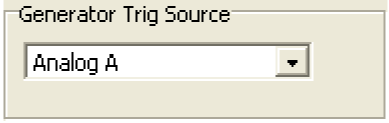
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Trigger.JitterLevel("pctFS")
SR1.Alyzr(0).Trigger.JitterLevel("pctFS") = 50
```

Description: Trigger level when the analyzer is set to Jitter Analyzer.


Generator Trig Source

Analog A

GenTriggerSource

Command Syntax: SR1.Alyzr(i).Trigger.GenTriggerSource

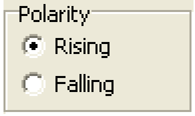
Command Argument(s):

Property: TrgGenSrc enumeration {trgAnlgA=0 | trgAnlgB=1 | trgDigA=2 | trgDigB=3 | trgJitter=4}

Example:

```
val = SR1.Alyzr(0).Trigger.GenTriggerSource
SR1.Alyzr(0).Trigger.GenTriggerSource = trgAnlgA
```

Description: Generator trigger source.


Polarity

● Rising

○ Falling

Polarity

Command Syntax: SR1.Alyzr(i).Trigger.Polarity

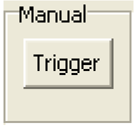
Command Argument(s):

Property: TrgPolarity enumeration {trgRising=0 | trgFalling=1}

Example:

```
val = SR1.Alyzr(0).Trigger.Polarity
SR1.Alyzr(0).Trigger.Polarity = trgRising
```

Description: Trigger polarity.



ManTrig

Command Syntax: SR1.Alyzr(i).Trigger.ManTrig()

Command Argument(s): None

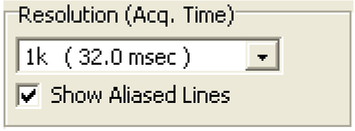
Example:

```
Call SR1.Alyzr(0).Trigger.ManTrig()
```

Description: Performs a manual trigger.

2.4.6.2 FFT1 Analyzer

Object:	**SR1.Alyzr(*i*).FFT**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the Single Channel FFT (FFT1) analyzer belonging to either A0 or A1.



Resolution (Acq. Time)

1k (32.0 msec)

☑ Show Aliased Lines

Lines

Command Syntax: SR1.Alyzr(i).FFT.Lines

Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:

```
val = SR1.Alyzr(0).FFT.Lines
SR1.Alyzr(0).FFT.Lines = fftl32k
```

Description: Number of FFT lines (resolution).

ShowAllLines

Command Syntax: SR1.Alyzr(i).FFT.ShowAllLines

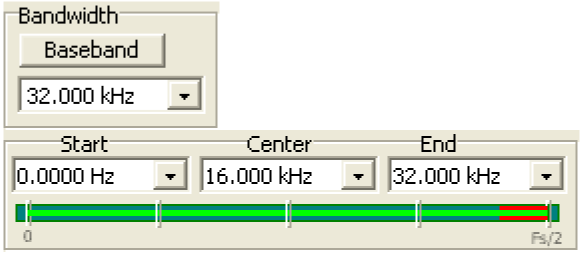
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT.ShowAllLines
SR1.Alyzr(0).FFT.ShowAllLines = False
```

Description: State of the "Show Aliased Lines" checkbox.



Bandwidth

32.000 kHz

Start: 0.0000 Hz

Center: 16.000 kHz

End: 32.000 kHz

Span

Command Syntax: SR1.Alyzr(i).FFT.Span

Command Argument(s):

Property: FFTbw enumeration {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}

Example:

```
val = SR1.Alyzr(0).FFT.Span
SR1.Alyzr(0).FFT.Span = fftFsDiv2
```

Description: Frequency span (bandwidth) setting of the FFT analyzer. The maximum span is Fs/2, represented by the enumeration fftFsDiv2. Each subsequent setting divides the frequency span by 2.

Baseband

Command Syntax: SR1.Alyzr(i).FFT.Baseband()

Command Argument(s): None

Example:

```
Call SR1.Alyzr(0).FFT.Baseband()
```

Description: Sets the FFT analyzer to its maximum frequency range.

StartFreq

Command Syntax: SR1.Alyzr(i).FFT.StartFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.StartFreq("Hz")
SR1.Alyzr(0).FFT.StartFreq("Hz") = 0
```

Description: Value of the lowest frequency in the analysis region.

CenterFreq

Command Syntax: SR1.Alyzr(i).FFT.CenterFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.CenterFreq("Hz")
SR1.Alyzr(0).FFT.CenterFreq("Hz") = 16000.0
```

Description: Value of the midpoint of the analysis region.

StopFreq

Command Syntax: SR1.Alyzr(i).FFT.StopFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.StopFreq("Hz")
SR1.Alyzr(0).FFT.StopFreq("Hz") = 32000
```

Description: Value of the highest frequency in the analysis region.

TimeRecordDuration

Command Syntax: SR1.Alyzr(i).FFT.TimeRecordDuration(*ValueUnit*)

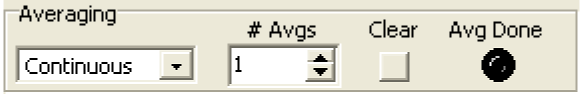
Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).FFT.TimeRecordDuration("s")
```

Description: Length of the time record for the current span and resolution settings.



Averaging

Command Syntax: SR1.Alyzr(i).FFT.Averaging

Command Argument(s):

Property: AvgModes enumeration {avgNone=0 | avgFixedLength=1 | avgContinuous=2}

Example:

```
val = SR1.Alyzr(0).FFT.Averaging
SR1.Alyzr(0).FFT.Averaging = avgNone
```

Description: Averaging mode for the FFT analyzer.

AvgDone

Command Syntax: SR1.Alyzr(i).FFT.AvgDone

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT.AvgDone
```

Description: "Average Done" status of the FFT1 analyzer.

NumAverages

Command Syntax: SR1.Alyzr(i).FFT.NumAverages

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).FFT.NumAverages
SR1.Alyzr(0).FFT.NumAverages = 3
```

Description: Number of averages for the linear or power spectrum.

ResetAvg

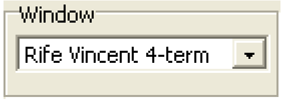
Command Syntax: SR1.Alyzr(i).FFT.ResetAvg()

Command Argument(s): None

Example:

```
Call SR1.Alyzr(0).FFT.ResetAvg()
```

Description: Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



Window: Rife Vincent 4-term

Window

Command Syntax: SR1.Alyzr(i).FFT.Window

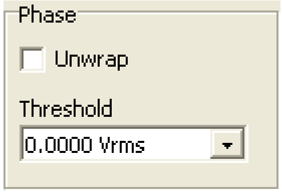
Command Argument(s):

Property: FFTwindows enumeration {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:

```
val = SR1.Alyzr(0).FFT.Window
SR1.Alyzr(0).FFT.Window = fftBlackmanHarris
```

Description: Window selection for the analyzer.



Phase

□ Unwrap

Threshold: 0.0000 Vrms

AnlgPhaseThreshold

Command Syntax: SR1.Alyzr(i).FFT.AnlgPhaseThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.AnlgPhaseThreshold("Vrms")
SR1.Alyzr(0).FFT.AnlgPhaseThreshold("Vrms") = 0.0001
```

Description: Amplitude threshold for calculating phase for analog inputs.

DigPhaseThreshold

Command Syntax: SR1.Alyzr(i).FFT.DigPhaseThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.DigPhaseThreshold("FFS")
SR1.Alyzr(0).FFT.DigPhaseThreshold("FFS") = 0.0001
```

Description: Amplitude threshold for calculating phase for digital audio inputs.

PhaseUnwrap

Command Syntax: SR1.Alyzr(i).FFT.PhaseUnwrap

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT.PhaseUnwrap
SR1.Alyzr(0).FFT.PhaseUnwrap = False
```

Description: Phase Unwrap status for the analyzer.

DC Correction

None

DCCorrectionMode

Command Syntax: SR1.Alyzr(i).FFT.DCCorrectionMode

Command Argument(s):

Property: dccorr enumeration {dcmNone=0 | dcmMean=1 | dcmPkPk=2}

Example:

```
val = SR1.Alyzr(0).FFT.DCCorrectionMode
SR1.Alyzr(0).FFT.DCCorrectionMode = dcmNone
```

Description: DC Correction mode setting for the analyzer.

Time Display Interpolation

○ Off ● On

DisplayInterpolation

Command Syntax: SR1.Alyzr(i).FFT.DisplayInterpolation

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT.DisplayInterpolation
SR1.Alyzr(0).FFT.DisplayInterpolation = False
```

Description: Time-domain display interpolation status for the analyzer.

Spectrum Weighting

None

□ Invert

SetEqFile

Command Syntax: SR1.Alyzr(i).FFT.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.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. `SR1.Alyzr(0).FFT.SetEqFile("")`

InvertEq

Command Syntax: SR1.Alyzr(i).FFT.InvertEq

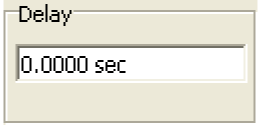
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT.InvertEq
SR1.Alyzr(0).FFT.InvertEq = False
```

Description: Invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.



Delay

0.0000 sec

TriggerDelay

Command Syntax: SR1.Alyzr(i).FFT.TriggerDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT.TriggerDelay("s")
SR1.Alyzr(0).FFT.TriggerDelay("s") = 0.0000325
```

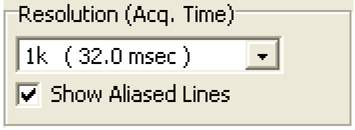
Description: Delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample, (1/Fs). Negative (pre-trigger) delays are allowed.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.3 FFT2 Analyzer

Object:	**SR1.Alyzr(*i*).FFT2Ch**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the Dual Channel FFT analyzer (FFT2) belonging to either A0 or A1.



Resolution (Acq. Time)

1k (32.0 msec)

☑ Show Aliased Lines

Lines

Command Syntax: SR1.Alyzr(i).FFT2Ch.Lines

Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:
```
val = SR1.Alyzr(0).FFT2Ch.Lines
SR1.Alyzr(0).FFT2Ch.Lines = fftl32k
```

Description: Number of FFT lines (resolution).

ShowAllLines

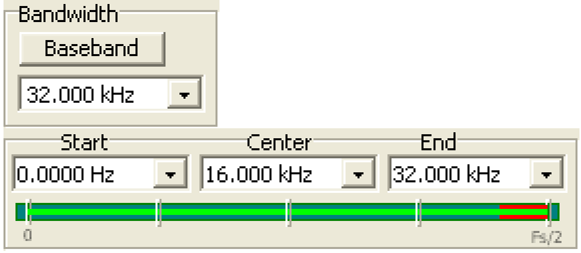
Command Syntax: SR1.Alyzr(i).FFT2Ch.ShowAllLines

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).FFT2Ch.ShowAllLines
SR1.Alyzr(0).FFT2Ch.ShowAllLines = False
```

Description: State of the "Show Aliased Lines" checkbox.



Bandwidth

32.000 kHz

Start: 0.0000 Hz

Center: 16.000 kHz

End: 32.000 kHz

Span

Command Syntax: SR1.Alyzr(i).FFT2Ch.Span

Command Argument(s):

Property: FFTbw enumeration {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.Span
SR1.Alyzr(0).FFT2Ch.Span = fftFsDiv2
```

Description: Frequency span(bandwidth) settling of the FFT2 analyzer. The maximum span is Fs/2, represented by the enumeration fftFsDiv2. Each subsequent setting divides the frequency span by 2.

Baseband

Command Syntax: SR1.Alyzr(i).FFT2Ch.Baseband()

Command Argument(s): None

Example:

```
Call SR1.Alyzr(0).FFT2Ch.Baseband()
```

Description: Sets the FFT2 analyzer frequency span to its maximum value.

StartFreq

Command Syntax: SR1.Alyzr(i).FFT2Ch.StartFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.StartFreq("Hz")
SR1.Alyzr(0).FFT2Ch.StartFreq("Hz") = 0
```

Description: Value of the lowest frequency in the analysis range.

CenterFreq

Command Syntax: SR1.Alyzr(i).FFT2Ch.CenterFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.CenterFreq("Hz")
SR1.Alyzr(0).FFT2Ch.CenterFreq("Hz") = 50000
```

Description: Value of the midpoint of the analysis range.

StopFreq

Command Syntax: SR1.Alyzr(i).FFT2Ch.StopFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.StopFreq("Hz")
SR1.Alyzr(0).FFT2Ch.StopFreq("Hz") = 200000
```

Description: Value of the highest frequency in the analysis range.

TimeRecordDuration

Command Syntax: SR1.Alyzr(i).FFT2Ch.TimeRecordDuration(*ValueUnit*)

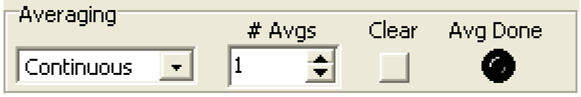
Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).FFT2Ch.TimeRecordDuration("s")
```

Description: Length of the time record for the current span and resolution settings.



Averaging

Command Syntax: SR1.Alyzr(i).FFT2Ch.Averaging

Command Argument(s):

Property: AvgModes enumeration {avgNone=0 | avgFixedLength=1 | avgContinuous=2}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.Averaging
SR1.Alyzr(0).FFT2Ch.Averaging = avgNone
```

Description: Averaging mode for the FFT2 analyzer.

AvgDone

Command Syntax: SR1.Alyzr(i).FFT2Ch.AvgDone

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.AvgDone
```

Description: "Average Done" status of the FFT2 analyzer.

NumAverages

Command Syntax: SR1.Alyzr(i).FFT2Ch.NumAverages

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Alyzr(0).FFT2Ch.NumAverages
SR1.Alyzr(0).FFT2Ch.NumAverages = 10
```

Description: Number of averages used by the FFT2 analyzer.

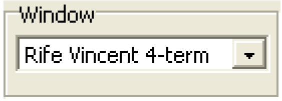
ResetAvg

Command Syntax: SR1.Alyzr(i).FFT2Ch.ResetAvg()

Command Argument(s): None

Example:
```
Call SR1.Alyzr(0).FFT2Ch.ResetAvg()
```

Description: Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



Window

Rife Vincent 4-term

Window

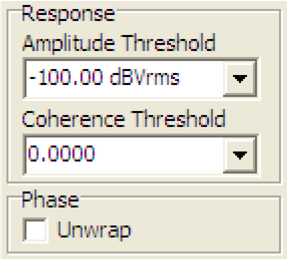
Command Syntax: SR1.Alyzr(i).FFT2Ch.Window

Command Argument(s):

Property: FFTwindows enumeration {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:
```
val = SR1.Alyzr(0).FFT2Ch.Window
SR1.Alyzr(0).FFT2Ch.Window = fftBlackmanHarris
```

Description: Window selection for the analyzer.



Response

Amplitude Threshold

-100.00 dBVrms

Coherence Threshold

0.0000

Phase

☐ Unwrap

CoherenceRespThreshold

Command Syntax: SR1.Alyzr(i).FFT2Ch.CoherenceRespThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.CoherenceRespThreshold("")
SR1.Alyzr(0).FFT2Ch.CoherenceRespThreshold("") = 0.5
```

Description: 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.

AnlgRespThreshold

Command Syntax: SR1.Alyzr(i).FFT2Ch.AnlgRespThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.AnlgRespThreshold("dBVrms")
SR1.Alyzr(0).FFT2Ch.AnlgRespThreshold("dBVrms") = -100
```

Description: Analog magnitude threshold for frequency response computation. The analyzer will report 0 magnitude and phase for any frequency for which which the A- or B-channel FFT magnitude is below this threshold.

DigRespThreshold

Command Syntax: SR1.Alyzr(i).FFT2Ch.DigRespThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.DigRespThreshold("dBFS")
SR1.Alyzr(0).FFT2Ch.DigRespThreshold("dBFS") = -120
```

Description: Digital magnitude threshold for frequency response computation. The analyzer will report 0 magnitude and phase for any frequency for which which the A- or B-channel FFT magnitude is below this threshold.

PhaseUnwrap

Command Syntax: SR1.Alyzr(i).FFT2Ch.PhaseUnwrap

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.PhaseUnwrap
SR1.Alyzr(0).FFT2Ch.PhaseUnwrap = False
```

Description: Phase unwrap setting for the FFT2 analyzer.

DC Correction

None

DCCorrectionMode

Command Syntax: SR1.Alyzr(i).FFT2Ch.DCCorrectionMode

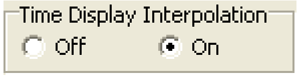
Command Argument(s):

Property: dccorr enumeration {dcmNone=0 | dcmMean=1 | dcmPkPk=2}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.DCCorrectionMode
SR1.Alyzr(0).FFT2Ch.DCCorrectionMode = dcmNone
```

Description: DC Correction mode setting for the FFT2 analyzer.



Time Display Interpolation

○ Off ● On

DisplayInterpolation

Command Syntax: SR1.Alyzr(i).FFT2Ch.DisplayInterpolation

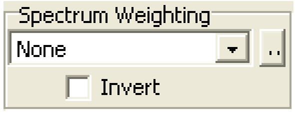
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.DisplayInterpolation
SR1.Alyzr(0).FFT2Ch.DisplayInterpolation = False
```

Description: Display interpolation mode for the FFT2 analyzer.



Spectrum Weighting

None

□ Invert

SetEqFile

Command Syntax: SR1.Alyzr(i).FFT2Ch.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.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.

```
SR1.Alyzr(0).FFT2ch.SetEqFile("")
```

InvertEq

Command Syntax: SR1.Alyzr(i).FFT2Ch.InvertEq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.InvertEq
SR1.Alyzr(0).FFT2Ch.InvertEq = False
```

Description: Invert EQ status of the FFT2 analyzer. If on, the spectrum is weighted by the inverse of the response given by the EQ file.

CalcImpulseResponse

Command Syntax: SR1.Alyzr(i).FFT2Ch.CalcImpulseResponse

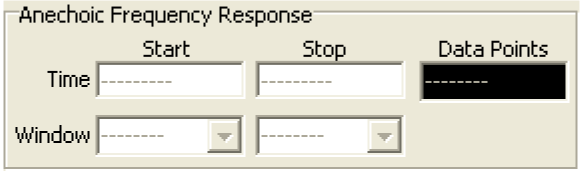
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.CalcImpulseResponse
SR1.Alyzr(0).FFT2Ch.CalcImpulseResponse = False
```

Description: Determines whether impulse response measurements will be calculated.



IRStart

Command Syntax: SR1.Alyzr(i).FFT2Ch.IRStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.IRStart("")
SR1.Alyzr(0).FFT2Ch.IRStart("") = -0.016
```

Description: 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: SR1.Alyzr(i).FFT2Ch.IRStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.IRStop("s")
SR1.Alyzr(0).FFT2Ch.IRStop("s") = 0.015984
```

Description: 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: SR1.Alyzr(i).FFT2Ch.IRStartWin

Property: ImpRespWindow enumeration {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.IRStartWin
SR1.Alyzr(0).FFT2Ch.IRStartWin = irwNone
```

Description: 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: SR1.Alyzr(i).FFT2Ch.IRStopWin

Command Argument(s):

Property: ImpRespWindow enumeration {irwNone=0 | irw5pct=1 | irw10pct=2 | irw20pct=3 | irw30pct=4}

Example:

```
val = SR1.Alyzr(0).FFT2Ch.IRStopWin
SR1.Alyzr(0).FFT2Ch.IRStopWin = irwNone
```

Description: 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: SR1.Alyzr(i).FFT2Ch.IRNumPoints

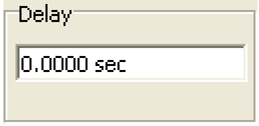
Command Argument(s):

Property: Integer (read-only)

Example:

```
val = SR1.Alyzr(0).FFT2Ch.IRNumPoints
```

Description: Number of points in the region selected for use in calculating the anechoic frequency response.



Delay

0.0000 sec

TriggerDelay

Command Syntax: SR1.Alyzr(i).FFT2Ch.TriggerDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).FFT2Ch.TriggerDelay("s")
SR1.Alyzr(0).FFT2Ch.TriggerDelay("s") = 0
```

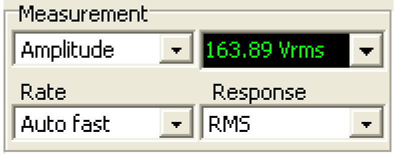
Description: Delay from the trigger to the beginning of the time record. The resolution of this control is 1 sample (1/Fs). Negative (pre-triggered) delays are allowed.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.4 Time Domain Detector

Object:	**SR1.Alyzr(*i*).TimeDomDetector**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands pertaining to the Time Domain Detector on either A0 or A1.



Measurement
Amplitude
163.89 Vrms
Rate: Auto fast
Response: RMS

Measurement

Command Syntax: SR1.Alyzr(i).TimeDomDetector.Measurement

Command Argument(s):

Property: DetectorMeas enumeration {adAmplitude=0 | adRatio=1 | adTHDNAmp=2 | adTHDNRatio=3 | adSINADRatio=4 | adCrestFactor=5}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.Measurement
SR1.Alyzr(0).TimeDomDetector.Measurement = adAmplitude
```

Description: Measurement being performed by the TDD analyzer.

DetectorRdg

Command Syntax: SR1.Alyzr(i).TimeDomDetector.DetectorRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.DetectorRdg("Vrms")
```

Description: Current measurement reading of the TDD.

DetectRate

Command Syntax: SR1.Alyzr(i).TimeDomDetector.DetectRate

Command Argument(s):

Property: DetectorRates enumeration {adFast=0 | adPrecise=1 | adrps1=2 | adrps2=3 | adrps4=4 | adrps8=5 | adrps16=6 | adrps32=7 | adrps64=8 | adrps128=9 | adrps256=10}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.DetectRate
SR1.Alyzr(0).TimeDomDetector.DetectRate = adFast
```

Description: Detector rate setting of the TDD analyzer.

Response

Command Syntax: SR1.Alyzr(i).TimeDomDetector.Response

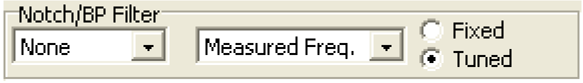
Command Argument(s):

Property: DetectorResponseTypes enumeration {adRMS=0 | adQuasiPk=1 | adPeak=2}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.Response
SR1.Alyzr(0).TimeDomDetector.Response = adRMS
```

Description: Response setting of the TDD.



Notch/BP Filter

None

Measured Freq.

○ Fixed

● Tuned

NotchBPFilt

Command Syntax: SR1.Alyzr(i).TimeDomDetector.NotchBPFilt

Command Argument(s):

Property: DetectorFiltMode enumeration {adNoNotchBP=0 | adNotch=1 | adBPHW=2 | adBP3rdOct=3 | adBP6thOct=4 | adBP12Oct=5 | adBP24Oct=6}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.NotchBPFilt
SR1.Alyzr(0).TimeDomDetector.NotchBPFilt = adNoNotchBP
```

Description: Notch/Bandpass setting for the TDD analyzer.

NotchBPFixedFreq

Command Syntax: SR1.Alyzr(i).TimeDomDetector.NotchBPFixedFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.NotchBPFixedFreq("Hz"
SR1.Alyzr(0).TimeDomDetector.NotchBPFixedFreq("Hz") = 10
```

Description: Fixed notch/bandpass center frequency when the notch bandpass tuning mode is set to fixed frequency.

NotchBPTuningMode

Command Syntax: SR1.Alyzr(i).TimeDomDetector.NotchBPTuningMode

Command Argument(s):

Property: TuningMode enumeration {tmFixed=0 | tmTuned=1}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.NotchBPTuningMode
SR1.Alyzr(0).TimeDomDetector.NotchBPTuningMode = tmFixed
```

Description: Notch/bandpass tuning mode.

NotchBPTuningSource

Command Syntax: SR1.Alyzr(i).TimeDomDetector.NotchBPTuningSource

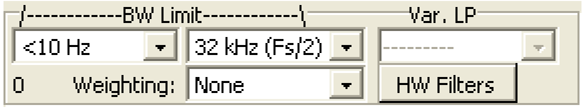
Command Argument(s):

Property: TuningSource enumeration {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.NotchBPTuningSource
SR1.Alyzr(0).TimeDomDetector.NotchBPTuningSource = tsMea
```

Description: Notch/bandpass tuning source when the tuning mode is set to "tuned."



<10 Hz | 32 kHz (Fs/2) | ---------

0 Weighting: None

BandEdgeLo

Command Syntax: SR1.Alyzr(i).TimeDomDetector.BandEdgeLo

Command Argument(s):

Property: DetectorHPfilt enumeration {adDC=0 | adHz22=1 | adHz100=2 | adHz400=3 | adSharp400=4 | adHzAES20=5 | adHzAES40=6 | adHzAES80=7}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.BandEdgeLo
SR1.Alyzr(0).TimeDomDetector.BandEdgeLo = adDC
```

Description: Setting of the low-frequency bandwidth limiting filter.

BandEdgeHi

Command Syntax: SR1.Alyzr(i).TimeDomDetector.BandEdgeHi

Command Argument(s):

Property: DetectorLPfilt enumeration {adFsdiv2=0 | adHzVar=1 | adHzAES20k=2 | adHzAES40k=3 | adHzAES80k=4}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.BandEdgeHi
SR1.Alyzr(0).TimeDomDetector.BandEdgeHi = adFsdiv2
```

Description: High-frequency bandwidth limiting filter mode.

BandEdgeHiVarFreq

Command Syntax: SR1.Alyzr(i).TimeDomDetector.BandEdgeHiVarFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.BandEdgeHiVarFreq("Hz
SR1.Alyzr(0).TimeDomDetector.BandEdgeHiVarFreq("Hz") = 2
```

Description: High-frequency filter cutoff when the high-frequency bandwidth limiting filter mode is set to "variable." (adHzVar)

WeightingFilt

Command Syntax: SR1.Alyzr(i).TimeDomDetector.WeightingFilt

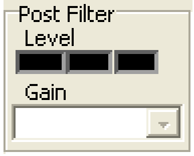
Command Argument(s):

Property: WeightingFilters enumeration {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.WeightingFilt
SR1.Alyzr(0).TimeDomDetector.WeightingFilt = adNoWt
```

Description: Weighting filter mode.



PostFilterGain

Command Syntax: SR1.Alyzr(i).TimeDomDetector.PostFilterGain

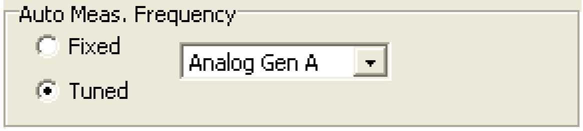
Command Argument(s):

Property: PostFiltGain enumeration {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:

```
val = SR1.Alyzr(0).TimeDomDetector.PostFilterGain
SR1.Alyzr(0).TimeDomDetector.PostFilterGain = pfgAuto
```

Description: Post-filter gain setting. Note that post filter gain is only active for analog inputs when using the HiBandwidth converter.


Auto Meas. Frequency

○ Fixed

● Tuned

Analog Gen A

AutoDetectMode

Command Syntax: SR1.Alyzr(i).TimeDomDetector.AutoDetectMode

Command Argument(s):

Property: TuningMode enumeration {tmFixed=0 | tmTuned=1}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.AutoDetectMode
SR1.Alyzr(0).TimeDomDetector.AutoDetectMode = tmFixed
```

Description: 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: SR1.Alyzr(i).TimeDomDetector.AutoDetect

Command Argument(s):

Property: TuningSource enumeration {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.AutoDetect
SR1.Alyzr(0).TimeDomDetector.AutoDetect = tsMeasFreq
```

Description: Source of the auto-detect frequency.

AutoDetectFreq

Command Syntax: SR1.Alyzr(i).TimeDomDetector.AutoDetectFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).TimeDomDetector.AutoDetectFreq("Hz")
SR1.Alyzr(0).TimeDomDetector.AutoDetectFreq("Hz") = 1000
```

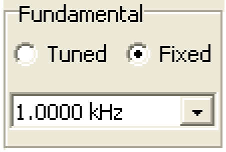
Description: Fixed frequency detect rate.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.5 THD Analyzer

Object:	**SR1.Alyzr(i).THD**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands pertaining to the THD analyzer on either A0 or A1.



Fundamental

○ Tuned ● Fixed

1.0000 kHz

TuningMode

Command Syntax: SR1.Alyzr(i).THD.TuningMode

Command Argument(s):

Property: TuningMode enumeration {tmFixed=0 | tmTuned=1}

Example:

```
val = SR1.Alyzr(0).THD.TuningMode
SR1.Alyzr(0).THD.TuningMode = tmFixed
```

Description: Fixed/tuned setting for the fundamental of the THD analyzer.

TuningSource

Command Syntax: SR1.Alyzr(i).THD.TuningSource

Command Argument(s):

Property: TuningSource enumeration {tsMeasFreq=0 | tsAnlgGenA=1 | tsAnlgGenB=2 | tsDigGenA=3 | tsDigGenB=4 | tsSweepSrc0=5 | tsSweepSrc1=6}

Example:

```
val = SR1.Alyzr(0).THD.TuningSource
SR1.Alyzr(0).THD.TuningSource = tsMeasFreq
```

Description: Fundamental tuning source when the tuning mode is "tuned."

FixedFreq

Command Syntax: SR1.Alyzr(i).THD.FixedFreq(*ValueUnit*)

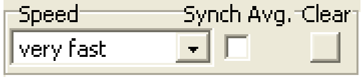
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).THD.FixedFreq("Hz")
SR1.Alyzr(0).THD.FixedFreq("Hz") = 1000
```

Description: Fundamental frequency when the tuning mode is set to "fixed."



Averaging

Command Syntax: SR1.Alyzr(i).THD.Averaging

Command Argument(s):

Property: AvgSpeed enumeration {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}

Example:

```
val = SR1.Alyzr(0).THD.Averaging
SR1.Alyzr(0).THD.Averaging = avgFast
```

Description: Averaging mode for the THD analyzer.

SyncAveraging

Command Syntax: SR1.Alyzr(i).THD.SyncAveraging

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).THD.SyncAveraging
SR1.Alyzr(0).THD.SyncAveraging = 1
```

Description: On/off status of synchronous averaging for the THD analyzer.

ResetAvg

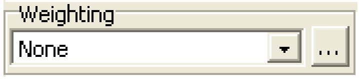
Command Syntax: SR1.Alyzr(i).THD.ResetAvg()

Command Argument(s): None

Example:

```
Call SR1.Alyzr(0).THD.ResetAvg()
```

Description: Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



SetEqFile

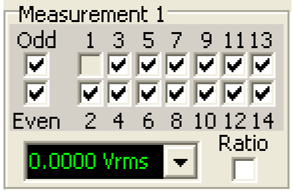
Command Syntax: SR1.Alyzr(i).THD.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.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. `SR1.Alyzr(0).FFT.SetEqFile("")`



Distortion0Rdg

Command Syntax: SR1.Alyzr(i).THD.Distortion0Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).THD.Distortion0Rdg("Vrms")`

Description: "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: SR1.Alyzr(i).THD.Distortion1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).THD.Distortion1Rdg("%")`

Description: "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: SR1.Alyzr(i).THD.MeasMode0

Command Argument(s):

Property: thdMeasMode enumeration {thdAmplitude=0 | thdRatio=1}

Example:

```
val = SR1.Alyzr(0).THD.MeasMode0
SR1.Alyzr(0).THD.MeasMode0 = thdAmplitude
```

Description: 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: SR1.Alyzr(i).THD.MeasMode1

Command Argument(s):

Property: thdMeasMode enumeration {thdAmplitude=0 | thdRatio=1}

Example:

```
val = SR1.Alyzr(0).THD.MeasMode1
SR1.Alyzr(0).THD.MeasMode1 = thdAmplitude
```

Description: 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: SR1.Alyzr(i).THD.SelectedHarmonics0

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Alyzr(0).THD.SelectedHarmonics0
SR1.Alyzr(0).THD.SelectedHarmonics0 = 12
```

Description: Harmonic selection for measurement 1 of the THD analyzer. The 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: SR1.Alyzr(i).THD.SelectedHarmonics1

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Alyzr(0).THD.SelectedHarmonics1
SR1.Alyzr(0).THD.SelectedHarmonics1 = 12
```

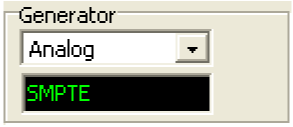
Description: See above.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.6 IMD Analyzer

Object:	**SR1.Alyzr(*i*).IMD**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands pertaining to the IMD analyzer on either A0 or A1.


Generator
Analog
SMPTE

GenLink

Command Syntax: SR1.Alyzr(i).IMD.GenLink

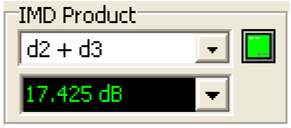
Command Argument(s):

Property: imdGenLinks enumeration {imdNoLink=-1 | imdDigGenA=0 | imdDigGenB=1 | imdAnlgGenA=2 | imdAnlgGenB=3}

Example:

```
val = SR1.Alyzr(0).IMD.GenLink
SR1.Alyzr(0).IMD.GenLink = imdNoLink
```

Description: Sets or 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.


IMD Product
d2 + d3
17.425 dB

DistortionRdg

Command Syntax: SR1.Alyzr(i).IMD.DistortionRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).IMD.DistortionRdg("dB")
```

Description: Current distortion reading of the IMD analyzer.

Product

Command Syntax: SR1.Alyzr(i).IMD.Product

Command Argument(s):

Property: imdProducts enumeration {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2},
{imdCCIF2=0 | imdCCIF23=1}, {imdDIMU5=0 | imdDIMU54=1}

Example:

```
val = SR1.Alyzr(0).IMD.Product
SR1.Alyzr(0).IMD.Product = imdSMPTE2
```

Description: IMD product selection for the IMD analyzer. Note the multiple enumerations due to the different IMD signals.

IsValid

Command Syntax: SR1.Alyzr(i).IMD.IsValid

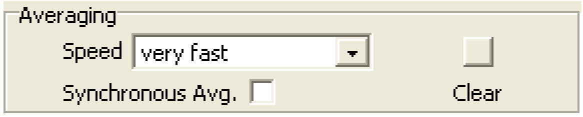
Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).IMD.IsValid
```

Description: 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

Speed very fast

Synchronous Avg. □

Averaging

Command Syntax: SR1.Alyzr(i).IMD.Averaging

Command Argument(s):

Property: AvgSpeed enumeration {avgFast=0 | avgOnedB=1 | avgHalfdB=2 | avgTenthdB=3}

Example:

```
val = SR1.Alyzr(0).IMD.Averaging
SR1.Alyzr(0).IMD.Averaging = avgFast
```

Description: Averaging selection for the IMD analyzer.

SyncAveraging

Command Syntax: SR1.Alyzr(i).IMD.SyncAveraging

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).IMD.SyncAveraging
SR1.Alyzr(0).IMD.SyncAveraging = False
```

Description: On/off status of synchronous averaging.

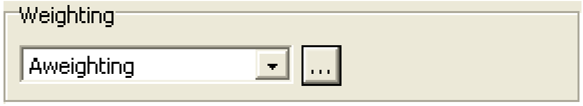
ResetAvg

Command Syntax: SR1.Alyzr(i).IMD.ResetAvg()

Command Argument(s): None

Example: `Call SR1.Alyzr(0).IMD.ResetAvg()`

Description: Resets the average buffer. This can be useful when using long averages to minimize the duration of transients.



Weighting

Aweighting

SetEqFile

Command Syntax: SR1.Alyzr(i).IMD.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Alyzr(0).IMD.SetEqFile()`

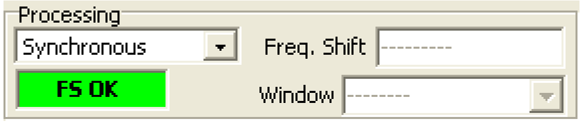
Description: Sets an EQ file to weight the IMD measurement. The argument is the file name, including the suffix. The default directory is "user\eqCurves". To remove spectrum weighting send an empty string enclosed in double quotes as the argument, i.e. `SR1.Alyzr(0).FFT.SetEqFile("")`

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.7 Multitone Analyzer

Object:	**SR1.Alyzr(*i*).Multitone**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the Multitone analyzer on either A0 or A1.



Processing: Synchronous

Freq. Shift: ---------

Window: --------

Processing

Command Syntax: SR1.Alyzr(i).Multitone.Processing

Command Argument(s):

Property: mtProc enumeration {mpSync=0 | mpWindow=1}

Example:

```
val = SR1.Alyzr(0).Multitone.Processing
SR1.Alyzr(0).Multitone.Processing = mpSync
```

Description: Processing mode (Synchronous or Windowed) for the multitone analyzer.

FreqShift

Command Syntax: SR1.Alyzr(i).Multitone.FreqShift(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Multitone.FreqShift("%")
SR1.Alyzr(0).Multitone.FreqShift("%") = 4
```

Description: Frequency shift setting (the maximum tone shift through the DUT) for the multitone analyzer.

Window

Command Syntax: SR1.Alyzr(i).Multitone.Window

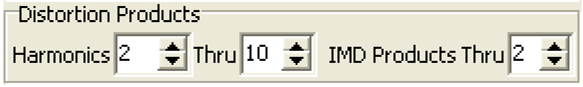
Command Argument(s):

Property: FFTwindows enumeration {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:

```
val = SR1.Alyzr(0).Multitone.Window
SR1.Alyzr(0).Multitone.Window = fftBlackmanHarris
```

Description: Window selection for the multitone analyzer running in windowed mode.



Distortion Products

Harmonics 2 Thru 10 IMD Products Thru 2

MaxHarmonic

Command Syntax: SR1.Alyzr(i).Multitone.MaxHarmonic

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).Multitone.MaxHarmonic
SR1.Alyzr(0).Multitone.MaxHarmonic = 10
```

Description: Maximum harmonic number analyzed by the multitone analyzer.

MaxIMDProduct

Command Syntax: SR1.Alyzr(i).Multitone.MaxIMDProduct

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).Multitone.MaxIMDProduct
SR1.Alyzr(0).Multitone.MaxIMDProduct = 2
```

Description: Order of the maxium IMD products computed by the multitone analyzer.

MinHarmonic

Command Syntax: SR1.Alyzr(i).Multitone.MinHarmonic

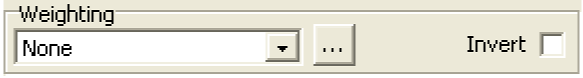
Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).Multitone.MinHarmonic
SR1.Alyzr(0).Multitone.MinHarmonic = 2
```

Description: Minimum harmonic number analyzed by the multitone analyzer.



Weighting

None

Invert □

SetEqFile

Command Syntax: SR1.Alyzr(i).Multitone.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Alyzr(0).Multitone.SetEqFile("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.
`SR1.Alyzr(0).Multitone.SetEqFile("")`

InvertEq

Command Syntax: SR1.Alyzr(i).Multitone.InvertEq

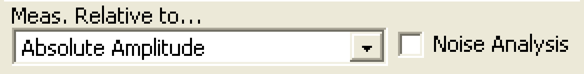
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Multitone.InvertEq
SR1.Alyzr(0).Multitone.InvertEq = False
```

Description: Invert EQ status of the analyzer. If On, the spectrum is weighted by the inverse of the specified EQ file response.



Meas. Relative to...

Absolute Amplitude

□ Noise Analysis

RelMode

Command Syntax: SR1.Alyzr(i).Multitone.RelMode

Command Argument(s):

Property: Boolean {mtAbsolute=0 | mtRelGen=1 | mtRelGenOther=2 | mtRelTone=3 | mtRelToneOther=4 | mtRelTotal=5}

Example:

```
val = SR1.Alyzr(0).Multitone.RelMode
SR1.Alyzr(0).Multitone.RelMode = mtAbsolute
```

Description: Relative mode used in computing the scalar measurements for the multitone analyzer.

AnalyzeNoise

Command Syntax: SR1.Alyzr(i).Multitone.AnalyzeNoise

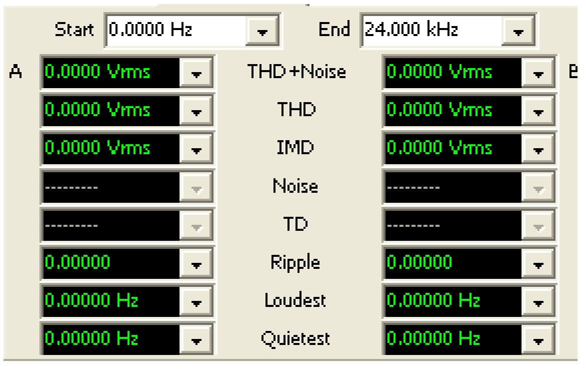
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Multitone.AnalyzeNoise
SR1.Alyzr(0).Multitone.AnalyzeNoise = False
```

Description: Status of the noise analysis checkbox. Noise analysis is only available in the synchronous processing. mode.



Start 0.0000 Hz End 24.000 kHz

A		B
0.0000 Vrms	THD+Noise	0.0000 Vrms
0.0000 Vrms	THD	0.0000 Vrms
0.0000 Vrms	IMD	0.0000 Vrms
---------	Noise	---------
---------	TD	---------
0.00000	Ripple	0.00000
0.00000 Hz	Loudest	0.00000 Hz
0.00000 Hz	Quietest	0.00000 Hz

StartFreq

Command Syntax: SR1.Alyzr(i).Multitone.StartFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).Multitone.StartFreq("Hz")
SR1.Alyzr(0).Multitone.StartFreq("Hz") = 0
```

Description: Minimum frequency used in the computation of the scalar measurements for the multitone analyzer.

StopFreq

Command Syntax: SR1.Alyzr(i).Multitone.StopFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).Multitone.StopFreq("Hz")
SR1.Alyzr(0).Multitone.StopFreq("Hz") = 24000.0
```

Description: Maximum frequency used in the computation of the scalar measurements for the multitone analyzer.

THDNARdg

Command Syntax: SR1.Alyzr(i).Multitone.THDNARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Alyzr(0).Multitone.THDNARdg("%")
```

Description: A channel THD_N measurement.

THDNBRdg

Command Syntax: SR1.Alyzr(i).Multitone.THDNBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Multitone.THDNBRdg("%")`

Description: B channel THD+N measurement.

THDARdg

Command Syntax: SR1.Alyzr(i).Multitone.THDARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Multitone.THDARdg("%")`

Description: A channel THD+N measurement.

THDBRdg

Command Syntax: SR1.Alyzr(i).Multitone.THDBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Multitone.THDBRdg("%")`

Description: B channel THD measurement.

IMDARdg

Command Syntax: SR1.Alyzr(i).Multitone.IMDARdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Multitone.IMDARdg("dB")`

Description: A channel IMD measurement.

IMDBRdg

Command Syntax: SR1.Alyzr(i).Multitone.IMDBRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Multitone.IMDBRdg("dB")`

Description: B channel IMD measurement.

NoiseARdg

Command Syntax: SR1.Alyzr(i).Multitone.NoiseARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.NoiseARdg("Vrms")`
Description: A channel noise measurement.

NoiseBRdg

Command Syntax: SR1.Alyzr(i).Multitone.NoiseBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.NoiseBRdg("Vrms")`
Description: B channel noise measurement.

TotalDistARdg

Command Syntax: SR1.Alyzr(i).Multitone.TotalDistARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.TotalDistARdg("%")`
Description: A channel total distortion measurement.

TotalDistBRdg

Command Syntax: SR1.Alyzr(i).Multitone.TotalDistBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.TotalDistBRdg("%")`
Description: B channel total distortion measurement.

RippleARdg

Command Syntax: SR1.Alyzr(i).Multitone.RippleARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.RippleARdg("%")`
Description: A channel total ripple measurement.

RippleBRdg

Command Syntax: SR1.Alyzr(i).Multitone.RippleBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.RippleBRdg("%")`
Description: B channel ripple measurement.

FreqMaxARdg

Command Syntax: SR1.Alyzr(i).Multitone.FreqMaxARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.FreqMaxARdg("Hz")`
Description: A channel loudest tone frequency..

FreqMaxBRdg

Command Syntax: SR1.Alyzr(i).Multitone.FreqMaxBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.FreqMaxBRdg("Hz")`
Description: B channel loudest tone frequency.

FreqMinARdg

Command Syntax: SR1.Alyzr(i).Multitone.FreqMinARdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.FreqMinARdg("Hz")`
Description: A channel quietest tone frequency.

FreqMinBRdg

Command Syntax: SR1.Alyzr(i).Multitone.FreqMinBRdg(*ValueUnit*)
Command Argument(s): *ValueUnit* as String
Property: Double (read-only)
Example: `val = SR1.Alyzr(0).Multitone.FreqMinBRdg("Hz")`
Description: B channel quietest tone frequency.

Delay
0.0000 sec

TriggerDelay

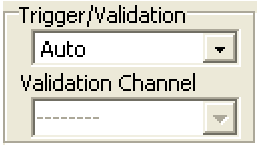
Command Syntax: SR1.Alyzr(i).Multitone.TriggerDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).Multitone.TriggerDelay("")
SR1.Alyzr(0).Multitone.TriggerDelay("") = Value
```

Description: 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: SR1.Alyzr(i).Multitone.TriggerMode

Command Argument(s):

Property: mtTrigger enumeration {mttAuto=0 | mttTrigger=1 | mttValLoose=2 | mttValNorm=3 | mttValTight=4}

Example:
```
val = SR1.Alyzr(0).Multitone.TriggerMode
SR1.Alyzr(0).Multitone.TriggerMode = mttAuto
```

Description: Trigger/validation mode for the multitone analyzer.

ValidationChannel

Command Syntax: SR1.Alyzr(i).Multitone.ValidationChannel

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Alyzr(0).Multitone.ValidationChannel
SR1.Alyzr(0).Multitone.ValidationChannel = 0
```

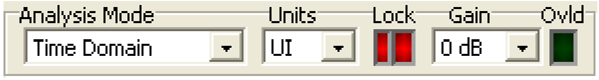
Description: Channel used for time-record validation.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.6.8 Jitter Analyzer

Object:	**SR1.Alyzr(*i*).Jitter**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands pertaining to the Jitter Analyzer



Analysis Mode: Time Domain

Units: UI

Lock

Gain: 0 dB

Ovld

Domain

Command Syntax: SR1.Alyzr(i).Jitter.Domain

Command Argument(s):

Property: jitdomain enumeration {jdTime=0 | jdFreq=1}

Example:

```
val = SR1.Alyzr(0).Jitter.Domain
SR1.Alyzr(0).Jitter.Domain = jdTime
```

Description: Domain, time or frequency, for the jitter analyzer.

MeasUnits

Command Syntax: SR1.Alyzr(i).Jitter.MeasUnits

Command Argument(s):

Property: JitterMeasUnits enumeration {muUI=0 | muSec=1}

Example:

```
val = SR1.Alyzr(0).Jitter.MeasUnits
SR1.Alyzr(0).Jitter.MeasUnits = muUI
```

Description: Jitter units that will be used by the analyzer..

ClockLockRdg

Command Syntax: SR1.Alyzr(i).Jitter.ClockLockRdg

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.ClockLockRdg
```

Description: Lock status of the digital audio receiver.

DemodLockRdg

Command Syntax: SR1.Alyzr(i).Jitter.DemodLockRdg

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.DemodLockRdg
```

Description: Lock status of the jitter demodulator.

ADCOverloadRdg

Command Syntax: SR1.Alyzr(i).Jitter.ADCOverloadRdg

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).Jitter.ADCOverloadRdg
```

Description: Overload status of the jitter analyzer.

Gain

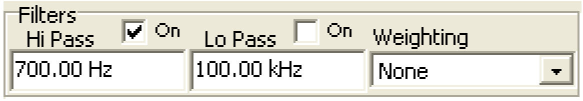
Command Syntax: SR1.Alyzr(i).Jitter.Gain

Command Argument(s):

Property: jitGain enumeration {jg1x=0 | jg10x=1|jg20x=2 | jg30x=3}

Example:
```
val = SR1.Alyzr(0).Jitter.Gain
SR1.Alyzr(0).Jitter.Gain = jg1x
```

Description: Jitter analyzer gain setting.



Filters

Hi Pass ☑ On: 700.00 Hz

Lo Pass □ On: 100.00 kHz

Weighting: None

TimeWeightingFilt

Command Syntax: SR1.Alyzr(i).Jitter.TimeWeightingFilt

Command Argument(s):

Property: WeightingFilters enumeration {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

Example:
```
val = SR1.Alyzr(0).Jitter.TimeWeightingFilt
SR1.Alyzr(0).Jitter.TimeWeightingFilt = adNoWt
```

Description: Weighting filter used by the jitter analyzer in time domain mode.

BandEdgeHi

Command Syntax: SR1.Alyzr(i).Jitter.BandEdgeHi(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).Jitter.BandEdgeHi("")
SR1.Alyzr(0).Jitter.BandEdgeHi("") = Value
```

Description: High frequency band-limiting filter frequency for time-domain mode.

BandEdgeHiOn

Command Syntax: SR1.Alyzr(i).Jitter.BandEdgeHiOn

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).Jitter.BandEdgeHiOn
SR1.Alyzr(0).Jitter.BandEdgeHiOn = False
```

Description: On/off status of the time-domain high frequency band-limiting filter.

BandEdgeLo

Command Syntax: SR1.Alyzr(i).Jitter.BandEdgeLo(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Alyzr(0).Jitter.BandEdgeLo("")
SR1.Alyzr(0).Jitter.BandEdgeLo("") = Value
```

Description: Low frequency band-limiting filter frequency for time-domain mode.

BandEdgeLoOn

Command Syntax: SR1.Alyzr(i).Jitter.BandEdgeLoOn

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).Jitter.BandEdgeLoOn
SR1.Alyzr(0).Jitter.BandEdgeLoOn = False
```

Description: On/off status of the time-domain low frequency band-limiting filter.

FreqSetEqFile

Command Syntax: SR1.Alyzr(i).Jitter.FreqSetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:
```
Call SR1.Alyzr(0).Jitter.FreqSetEqFile("Value")
```

Description: Sets the frequency-domain weighting filter.

InvertEq

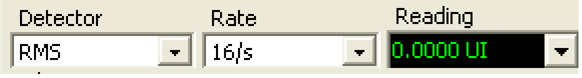
Command Syntax: SR1.Alyzr(i).Jitter.InvertEq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).Jitter.InvertEq
SR1.Alyzr(0).Jitter.InvertEq = False
```

Description: Invert/non-invert status of the frequency-domain weighting filter.


Detector: RMS
Rate: 16/s
Reading: 0.0000 UI

TimeResponse

Command Syntax: SR1.Alyzr(i).Jitter.TimeResponse

Command Argument(s):

Property: jtResponse enumeration {jtRMS=0 | jtPeak=1}

Example:

```
val = SR1.Alyzr(0).Jitter.TimeResponse
SR1.Alyzr(0).Jitter.TimeResponse = jtRMS
```

Description: Time-domain response type (rms or peak) of the jitter analyzer.

TimeDetectRate

Command Syntax: SR1.Alyzr(i).Jitter.TimeDetectRate

Command Argument(s):

Property: jtDetectRate enumeration {jtDigFs=0 | jtJitGen=1 | jtrps1=2 | jtrps2=3 | jtrps4=4 | jtrps8=5 | jtrps16=6 | jtrps32=7 | jtrps64=8 | jtrps128=9 | jtrps256=10}

Example:

```
val = SR1.Alyzr(0).Jitter.TimeDetectRate
SR1.Alyzr(0).Jitter.TimeDetectRate = jtDigFs
```

Description: Time-domain measurement rate.

TimeJitterRdg

Command Syntax: SR1.Alyzr(i).Jitter.TimeJitterRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).Jitter.TimeJitterRdg("")
```

Description: Jitter measured by the time-domain jitter analyzer.

Resolution: 4k
Bandwidth: 114.00 kHz
□ Show Aliased

FreqLines

Command Syntax: SR1.Alyzr(i).Jitter.FreqLines

Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqLines
SR1.Alyzr(0).Jitter.FreqLines = fftl32k
```

Description: Number of FFT lines used by the frequency-domain jitter analyzer.

FreqNumSpansDown

Command Syntax: SR1.Alyzr(i).Jitter.FreqNumSpansDown

Command Argument(s):

Property: FFTbw enumeration {fftFsDiv2=0 | fftFsDiv4=1 | fftFsDiv8=2 | fftFsDiv16=3 | fftFsDiv32=4 | fftFsDiv64=5 | fftFsDiv128=6 | fftFsDiv256=7 | fftFsDiv512=8 | fftFsDiv1024=9}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqNumSpansDown
SR1.Alyzr(0).Jitter.FreqNumSpansDown = fftFsDiv2
```

Description: Frequency range for the frequency-domain jitter analyzer. The largest frequency range corresponds to Fs/2, each subsequent range is x2 smaller.

FreqTimeRecordDuration

Command Syntax: SR1.Alyzr(i).Jitter.FreqTimeRecordDuration(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).Jitter.FreqTimeRecordDuration("s")
```

Description: Jitter time-record duration corresponding to the currently set frequency span.

FreqShowAllLines

Command Syntax: SR1.Alyzr(i).Jitter.FreqShowAllLines

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqShowAllLines
SR1.Alyzr(0).Jitter.FreqShowAllLines = False
```

Description: 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.

Window	Phase Thresh.	Unwrap	RMS Sum
Rife Vincent 4	100.00 mUI	□	0.0000 UI

FreqWindow

Command Syntax: SR1.Alyzr(i).Jitter.FreqWindow

Command Argument(s):

Property: FFTwindows enumeration {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:

```
val = SR1.Alyzr(0).Jitter.FreqWindow
SR1.Alyzr(0).Jitter.FreqWindow = fftBlackmanHarris
```

Description: Window used by frequency-domain jitter analyzer.

FreqPhaseThreshold

Command Syntax: SR1.Alyzr(i).Jitter.FreqPhaseThreshold(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Jitter.FreqPhaseThreshold("")
SR1.Alyzr(0).Jitter.FreqPhaseThreshold("") = Value
```

Description: Amplitude threshold for calculating jitter phase.

FreqPhaseUnwrap

Command Syntax: SR1.Alyzr(i).Jitter.FreqPhaseUnwrap

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqPhaseUnwrap
SR1.Alyzr(0).Jitter.FreqPhaseUnwrap = False
```

Description: On/off status of jitter phase unwrap.

FreqRmsJitter

Command Syntax: SR1.Alyzr(i).Jitter.FreqRmsJitter(*ValueUnit*)

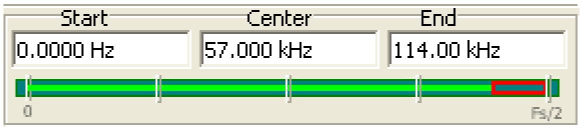
Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Alyzr(0).Jitter.FreqRmsJitter("")
```

Description: Total jitter calculated by the frequency domain jitter analyzer.



Start: 0.0000 Hz
Center: 57.000 kHz
End: 114.00 kHz

FreqStart

Command Syntax: SR1.Alyzr(i).Jitter.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Jitter.FreqStart("")
SR1.Alyzr(0).Jitter.FreqStart("") = 0.0
```

Description: Start frequency of the analysis range for the frequency-domain jitter analyzer.

FreqCenter

Command Syntax: SR1.Alyzr(i).Jitter.FreqCenter(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Jitter.FreqCenter("")
SR1.Alyzr(0).Jitter.FreqCenter("") = Value
```

Description: Center frequency of the analysis range for the frequency-domain jitter analyzer.

FreqStop

Command Syntax: SR1.Alyzr(i).Jitter.FreqStop(*ValueUnit*)

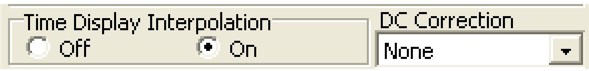
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Jitter.FreqStop("")
SR1.Alyzr(0).Jitter.FreqStop("") = Value
```

Description: Stop frequency of the analysis range for the frequency-domain jitter analyzer



Time Display Interpolation ○ Off ● On

DC Correction None

FreqDisplayInterpolation

Command Syntax: SR1.Alyzr(i).Jitter.FreqDisplayInterpolation

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqDisplayInterpolation
SR1.Alyzr(0).Jitter.FreqDisplayInterpolation = False
```

Description: On/off status of display interpolation for the frequency-domain jitter analyzer.

FreqDCCorrectionMode

Command Syntax: SR1.Alyzr(i).Jitter.FreqDCCorrectionMode

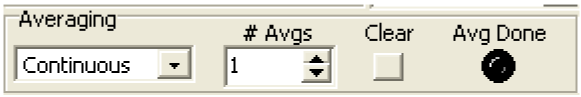
Command Argument(s):

Property: jitterDCCorr enumeration {jdcNone=0 | jdcAvg=1 | jdcHalfPkPk=2}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqDCCorrectionMode
SR1.Alyzr(0).Jitter.FreqDCCorrectionMode = jdcNone
```

Description: DC correction mode for the frequency-domain jitter analyzer.



FreqAveraging

Command Syntax: SR1.Alyzr(i).Jitter.FreqAveraging

Command Argument(s):

Property: AvgModes enumeration {avgNone=0 | avgFixedLength=1 | avgContinuous=2 | avgPeakFixedLength=3 | avgPeakContinuous=4}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqAveraging
SR1.Alyzr(0).Jitter.FreqAveraging = avgNone
```

Description: Averaging mode for the frequency domain jitter analyzer.

FreqAvgDone

Command Syntax: SR1.Alyzr(i).Jitter.FreqAvgDone

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Alyzr(0).Jitter.FreqAvgDone
```

Description: "Done" status for frequency-domain jitter averaging.

FreqNumAverages

Command Syntax: SR1.Alyzr(i).Jitter.FreqNumAverages

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Alyzr(0).Jitter.FreqNumAverages
SR1.Alyzr(0).Jitter.FreqNumAverages = 10
```

Description: Number of averages performed the analyzer.

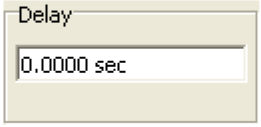
ResetAvg

Command Syntax: SR1.Alyzr(i).Jitter.ResetAvg()

Command Argument(s): None

Example: `Call SR1.Alyzr(0).Jitter.ResetAvg()`

Description: Resets the average buffer of the analyzer.



Delay

0.0000 sec

FreqTriggerDelay

Command Syntax: SR1.Alyzr(i).Jitter.FreqTriggerDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Jitter.FreqTriggerDelay("s")
SR1.Alyzr(0).Jitter.FreqTriggerDelay("s") = 0.016
```

Description: Delay between the trigger and the start of the time record for the frequency-domain jitter analyzer.

Form Commands:

OpenForm

OpenFormwID

CloseForm

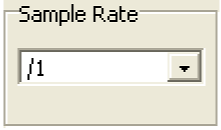
CloseForms

FormCount

FormID

2.4.6.9 Histogram Analyzer

Object:	**SR1.Alyzr(*i*).Histogram**
Object Argument(s):	*i* as Integer {0 | 1}
Description:	Commands related to the histogram analyzer on A0 or A1.



Sample Rate

/1

FsDivider

Command Syntax: SR1.Alyzr(i).Histogram.FsDivider

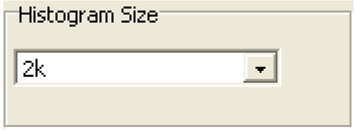
Command Argument(s):

Property: HistoDiv enumeration {hdFsdiv1=0 | hdFsdiv2=1 | hdFsdiv4=2 | hdFsdiv8=3 | hdFsdiv16=4 | hdFsdiv32=5 | hdFsdiv64=6 | hdFsdiv128=7 | hdFsdiv256=8 | hdFsdiv512=9}

Example:

```
val = SR1.Alyzr(0).Histogram.FsDivider
SR1.Alyzr(0).Histogram.FsDivider = hdFsdiv1
```

Description: Value of the Fs divider. If the divider is set to "/4", for example, the analyzer only examines every 4th point in the input data stream.



Histogram Size

2k

SampleSize

Command Syntax: SR1.Alyzr(i).Histogram.SampleSize

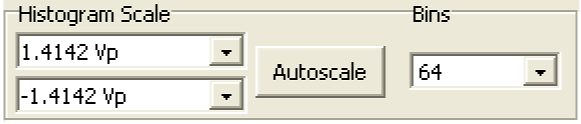
Command Argument(s):

Property: HistoSize enumeration {hs64k=0 | hs32k=1 | hs16k=2 | hs8k=3 | hs4k=4 | hs2k=5 | hs1k=6 | hs512=7}

Example:

```
val = SR1.Alyzr(0).Histogram.SampleSize
SR1.Alyzr(0).Histogram.SampleSize = hs64k
```

Description: Number of input points examined for each histogram.



Histogram Scale

1.4142 Vp

-1.4142 Vp

Bins

64

AnlgScaleMax

Command Syntax: SR1.Alyzr(i).Histogram.AnlgScaleMax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Histogram.AnlgScaleMax("")
SR1.Alyzr(0).Histogram.AnlgScaleMax("") = Value
```

Description: Input value corresponding to the maximum bin (for Analog inputs).

AnlgScaleMin

Command Syntax: SR1.Alyzr(i).Histogram.AnlgScaleMin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Histogram.AnlgScaleMin("")
SR1.Alyzr(0).Histogram.AnlgScaleMin("") = Value
```

Description: Input value corresponding to the minimum bin (for Analog inputs).

DigScaleMax

Command Syntax: SR1.Alyzr(i).Histogram.DigScaleMax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Histogram.DigScaleMax("")
SR1.Alyzr(0).Histogram.DigScaleMax("") = Value
```

Description: Input value corresponding to the maximum bin (for Digital Audio inputs).

DigScaleMin

Command Syntax: SR1.Alyzr(i).Histogram.DigScaleMin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Alyzr(0).Histogram.DigScaleMin("")
SR1.Alyzr(0).Histogram.DigScaleMin("") = Value
```

Description: Input value corresponding to the minimum bin (for Digital Audio inputs).

AutoScale

Command Syntax: SR1.Displays.Histogram.AutoScale()

Command Argument(s): None

Example: `Call SR1.Displays.Histogram.AutoScale()`

Description: Sets the Scale Min/Max to the minimum and maximum value of the current input data.

NumBins

Command Syntax: SR1.Alyzr(i).Histogram.NumBins

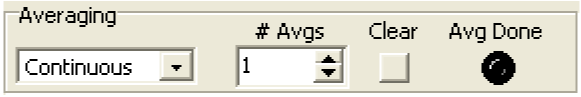
Command Argument(s):

Property: HistoBins enumeration {hb16=0 | hb32=1 | hb64=2 | hb128=3 | hb256=4 | hb512=5}

Example:

```
val = SR1.Alyzr(0).Histogram.NumBins
SR1.Alyzr(0).Histogram.NumBins = hb16
```

Description: Number of histogram bins.



Averaging: Continuous

Avgs: 1

Clear: □

Avg Done: ●

Averaging

Command Syntax: SR1.Alyzr(i).Histogram.Averaging

Command Argument(s):

Property: HistoAvg enumeration {haContinuous=0 | haSingle=1}

Example:

```
val = SR1.Alyzr(0).Histogram.Averaging
SR1.Alyzr(0).Histogram.Averaging = haContinuous
```

Description: Averaging type.

AvgDone

Command Syntax: SR1.Alyzr(i).Histogram.AvgDone

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example: `val = SR1.Alyzr(0).Histogram.AvgDone`

Description: "Average done" status.

NumAverages

Command Syntax: SR1.Alyzr(i).Histogram.NumAverages

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Alyzr(0).Histogram.NumAverages
SR1.Alyzr(0).Histogram.NumAverages = 10
```

Description: Number of averages.

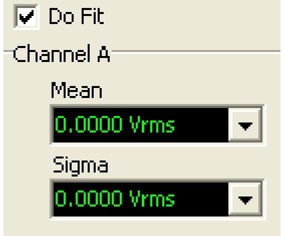
ResetAvg

Command Syntax: SR1.Alyzr(i).Histogram.ResetAvg()

Command Argument(s): None

Example:
```
Call SR1.Alyzr(0).Histogram.ResetAvg()
```

Description: Resets the average buffer.



☑ Do Fit

Channel A

Mean

0.0000 Vrms

Sigma

0.0000 Vrms

DoFit

Command Syntax: SR1.Alyzr(i).Histogram.DoFit

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Alyzr(0).Histogram.DoFit
SR1.Alyzr(0).Histogram.DoFit = False
```

Description: On/off status of the exponential fit.

MeanA

Command Syntax: SR1.Alyzr(i).Histogram.MeanA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Alyzr(0).Histogram.MeanA("Vp")
```

Description: Average (mean) value of the A channel input data.

MeanB

Command Syntax: SR1.Alyzr(i).Histogram.MeanB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Histogram.MeanB("Vp")`

Description: Average (mean) value of the A channel input data.

StdDevA

Command Syntax: SR1.Alyzr(i).Histogram.StdDevA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Histogram.StdDevA("")`

Description: Standard deviation of the A channel input data.

StdDevB

Command Syntax: SR1.Alyzr(i).Histogram.StdDevB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Alyzr(0).Histogram.StdDevB("")`

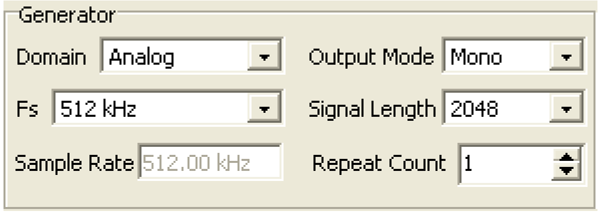
Description: Standard deviation of the B channel input data..

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.7 Multitone Configuration

Object:	**SR1.MultiToneCfg**
Object Argument(s):	None
Description:	Commands related to the configuration of the multitone signal (analog or digital).



Domain: Analog
Output Mode: Mono
Fs: 512 kHz
Signal Length: 2048
Sample Rate: 512.00 kHz
Repeat Count: 1

Generator

Command Syntax: SR1.MultiToneCfg.Generator

Command Argument(s):

Property: MTGen enumeration {Anlg=0 | Dig=1}

Example:

```
val = SR1.MultiToneCfg.Generator
SR1.MultiToneCfg.Generator = Anlg
```

Description: Generator selection, analog or digital for multitone analysis.

Stereo

Command Syntax: SR1.MultiToneCfg.Stereo

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.MultiToneCfg.Stereo
SR1.MultiToneCfg.Stereo = False
```

Description: Stereo/mono selection for multitone generation and analysis.

Fs

Command Syntax: SR1.MultiToneCfg.Fs

Command Argument(s):

Property: MTSR enumeration {mtHz512k=0 | mtHz128k=1 | mtHz64k=2 | mtISR=3 | mtOSR=4}

Example:

```
val = SR1.MultiToneCfg.Fs
SR1.MultiToneCfg.Fs = mtHz512k
```

Description: Generator sample rate mode for multitone generation.

Length

Command Syntax: SR1.MultiToneCfg.Length

Command Argument(s):

Property: MTLength enumeration {ml512=0 | ml1k=1 | ml2k=2 | ml4k=3 | ml8k=4 | ml16k=5 | ml32k=6}

Example:

```
val = SR1.MultiToneCfg.Length
SR1.MultiToneCfg.Length = ml512
```

Description: Length of the multitone signal in samples.

SampleRate

Command Syntax: SR1.MultiToneCfg.SampleRate(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.SampleRate("Hz")
SR1.MultiToneCfg.SampleRate("Hz") = 128000.0
```

Description: Actual value of the multitone generator sample rate.

RepeatCount

Command Syntax: SR1.MultiToneCfg.RepeatCount

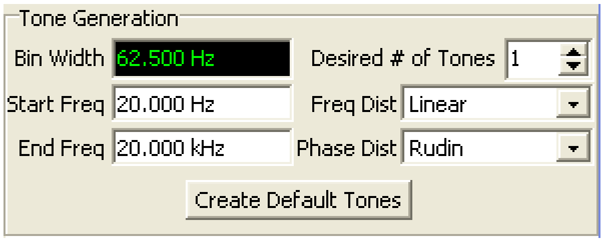
Command Argument(s):

Property: Integer

Example:

```
val = SR1.MultiToneCfg.RepeatCount
SR1.MultiToneCfg.RepeatCount = 4
```

Description: Generator repeat count. The repeat count determines how many repetitions of the multitone signal will occur in each burst.



Tone Generation

Bin Width 62.500 Hz

Desired # of Tones 1

Start Freq 20.000 Hz

Freq Dist Linear

End Freq 20.000 kHz

Phase Dist Rudin

BinWidthRdg

Command Syntax: SR1.MultiToneCfg.BinWidthRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.MultiToneCfg.BinWidthRdg("Hz")
```

Description: Minimum tone separation determined by the generator sample rate and signal length.

NumTonesNominal

Command Syntax: SR1.MultiToneCfg.NumTonesNominal

Command Argument(s):

Property: Integer

Example:

```
val = SR1.MultiToneCfg.NumTonesNominal
SR1.MultiToneCfg.NumTonesNominal = Value
```

Description: Desired number of tones.

StartFreq

Command Syntax: SR1.MultiToneCfg.StartFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.StartFreq("")
SR1.MultiToneCfg.StartFreq("") = Value
```

Description: Minimum tone frequency.

StopFreq

Command Syntax: SR1.MultiToneCfg.StopFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.StopFreq("")
SR1.MultiToneCfg.StopFreq("") = Value
```

Description: Maximum tone frequency.

FreqDist

Command Syntax: SR1.MultiToneCfg.FreqDist

Command Argument(s):

Property: MTFreqDist enumeration {mfLinear=0 | mfLog=1 | mfOctave=2 | mfDecade=3 | mfPrime=4 | mfLinearPrime=5 | mfLogPrime=6}

Example:

```
val = SR1.MultiToneCfg.FreqDist
SR1.MultiToneCfg.FreqDist = mfLinear
```

Description: Tone distribution algorithm.

PhaseDist

Command Syntax: SR1.MultiToneCfg.PhaseDist

Command Argument(s):

Property: MTPhaseDist enumeration {mpZero=0 | mpNewman=1 | mpSchroeder=2 | mpZygmund=3 | mpRudin=4 | mpRandom=5}

Example:
```
val = SR1.MultiToneCfg.PhaseDist
SR1.MultiToneCfg.PhaseDist = mpZero
```

Description: Phase distribution algorithm.

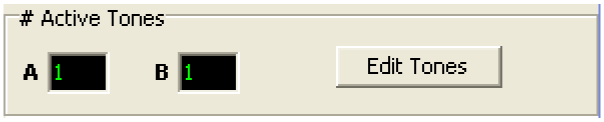
DefaultTones

Command Syntax: SR1.MultiToneCfg.DefaultTones()

Command Argument(s): None

Example:
```
Call SR1.MultiToneCfg.DefaultTones()
```

Description: Calculates a set of tones from the specified frequency limits and frequency and phase distributions.



A: 1

B: 1

NumTonesActualA

Command Syntax: SR1.MultiToneCfg.NumTonesActualA

Command Argument(s):

Property: Integer (read-only)

Example:
```
val = SR1.MultiToneCfg.NumTonesActualA
```

Description: Actual number of A-channel tones.

NumTonesActualB

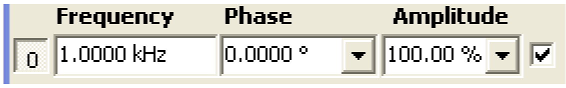
Command Syntax: SR1.MultiToneCfg.NumTonesActualB

Command Argument(s):

Property: Integer (read-only)

Example:
```
val = SR1.MultiToneCfg.NumTonesActualB
```

Description: Actual number of B-channel tones.



	Frequency	Phase	Amplitude	
0	1.0000 kHz	0.0000 °	100.00 %	☑

RelToneNumA

Command Syntax: SR1.MultiToneCfg.RelToneNumA

Command Argument(s):

Property: Integer

Example:
```
val = SR1.MultiToneCfg.RelToneNumA
SR1.MultiToneCfg.RelToneNumA = Value
```

Description: Index of the A-channel tone that will be used for tone-relative calculations.

RelToneNumB

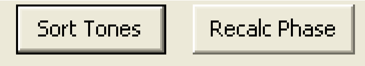
Command Syntax: SR1.MultiToneCfg.RelToneNumB

Command Argument(s):

Property: Integer

Example:
```
val = SR1.MultiToneCfg.RelToneNumB
SR1.MultiToneCfg.RelToneNumB = Value
```

Description: Index of the B-channel tone that will be used for tone-relative calculations.



OptimizePhase

Command Syntax: SR1.MultiToneCfg.OptimizePhase(*Channel*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}

Example: `Call SR1.MultiToneCfg.OptimizePhase()`

Description: Calculates the tone phases based on the selected phase distribution.

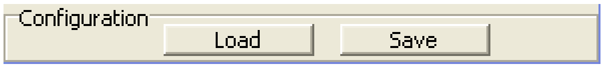
SortTones

Command Syntax: SR1.MultiToneCfg.SortTones(*Channel*)

Command Argument(s): *Channel* as ChanEnum enumeration {chanA=0 | chanB=1}

Example: `Call SR1.MultiToneCfg.SortTones()`

Description: Sorts the tones in order of increasing frequency.



Load

Command Syntax: SR1.MultiToneCfg.Load(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.MultiToneCfg.Load("Myfile.xml")`

Description: Loads the specified multitone configuration file.

Save

Command Syntax: SR1.MultiToneCfg.Save(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.MultiToneCfg.Save("Myfile.xml")`

Description: Saves the current multitone configuration to a file.

CloseTonesForm

Command Syntax: SR1.MultiToneCfg.CloseTonesForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.MultiToneCfg.CloseTonesForm(21)`

Description: Close the tone form with the specified FormID.

CloseTonesForms

Command Syntax: SR1.MultiToneCfg.CloseTonesForms()

Command Argument(s): None

Example: `Call SR1.MultiToneCfg.CloseTonesForms()`

Description: Close all tone forms on all pages of the page control.

OpenTonesForm

Command Syntax: SR1.MultiToneCfg.OpenTonesForm()

Command Argument(s): None

Example: `Call SR1.MultiToneCfg.OpenTonesForm()`

Description: Open a tone form on the current page of the page control.

OpenTonesFormwID

Command Syntax: SR1.MultiToneCfg.OpenTonesFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.MultiToneCfg.OpenTonesFormwID()`

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: SR1.MultiToneCfg.TonesFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.MultiToneCfg.TonesFormCount()`

Description: Total number of tone forms found on all pages of the page control.

TonesFormID

Command Syntax: SR1.MultiToneCfg.TonesFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.MultiToneCfg.TonesFormID(0)`

Description: Returns the FormID of the ith tone form.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.7.1 ToneA

Object:	**SR1.MultiToneCfg.ToneA(*i*)**
Object Argument(s):	*i* as Integer {0 | 1 | 2 | 3 | ... | 48 | 49}
Description:	Commands related to the ith A ch Multitone tone.

Amplitude

Command Syntax: SR1.MultiToneCfg.ToneA(i).Amplitude(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.ToneA(30).Amplitude("pct")
SR1.MultiToneCfg.ToneA(30).Amplitude("pct") = 80
```

Description: Amplitude of the tone. Tone amplitudes are specified in relative units.

Freq

Command Syntax: SR1.MultiToneCfg.ToneA(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.ToneA(3).Freq("Hz")
SR1.MultiToneCfg.ToneA(3).Freq("Hz") = 1000
```

Description: Frequency of the tone..

On

Command Syntax: SR1.MultiToneCfg.ToneA(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.MultiToneCfg.ToneA(37).On
SR1.MultiToneCfg.ToneA(37).On = False
```

Description: On/off status of the tone.

Phase

Command Syntax: SR1.MultiToneCfg.ToneA(i).Phase(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.MultiToneCfg.ToneA(45).Phase("deg")
SR1.MultiToneCfg.ToneA(45).Phase("deg") = 22
```

Description: Phase of the tone. Phase is specified for a *cosine* wave.

2.4.7.2 ToneB

Object:	**SR1.MultiToneCfg.ToneB(*i*)**
Object Argument(s):	*i* as Integer {0 | 1 | 2 | 3 | ... | 48 | 49}
Description:	Commands related to the i^th^ B ch Multitone tone.

Amplitude

Command Syntax: SR1.MultiToneCfg.ToneB(i).Amplitude(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.MultiToneCfg.ToneB(30).Amplitude("pct")
SR1.MultiToneCfg.ToneB(30).Amplitude("pct") = 80
```

Description: Amplitude of the tone. Tone amplitudes are specified in relative units.

Freq

Command Syntax: SR1.MultiToneCfg.ToneB(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.MultiToneCfg.ToneB(28).Freq("Hz")
SR1.MultiToneCfg.ToneB(28).Freq("Hz") = 1000
```

Description: Frequency of the tone.

On

Command Syntax: SR1.MultiToneCfg.ToneB(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.MultiToneCfg.ToneB(19).On
SR1.MultiToneCfg.ToneB(19).On = False
```

Description: On/off status of the tone.

Phase

Command Syntax: SR1.MultiToneCfg.ToneB(i).Phase(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

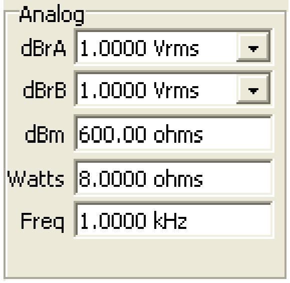
Property: Double

Example:
```
val = SR1.MultiToneCfg.ToneB(5).Phase("deg")
SR1.MultiToneCfg.ToneB(5).Phase("deg") = 22
```

Description: Phase of the tone. Phase is specified for a *cosine* wave.

2.4.8 Analyzer References

Object:	**SR1.AlyzrReferences**
Object Argument(s):	None
Description:	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: SR1.AlyzrReferences.AnlgdBrA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.AnlgdBrA("Vrms")
SR1.AlyzrReferences.AnlgdBrA("Vrms") = 1
```

Description: Analog dBrA reference.

AnlgdBrB

Command Syntax: SR1.AlyzrReferences.AnlgdBrB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.AnlgdBrB("Vrms")
SR1.AlyzrReferences.AnlgdBrB("Vrms") = 1
```

Description: Analog dBrB reference.

AnlgFreq

Command Syntax: SR1.AlyzrReferences.AnlgFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.AnlgFreq("Hz")
SR1.AlyzrReferences.AnlgFreq("Hz") = 1000.0
```

Description: Analog frequency reference.

dBmZ

Command Syntax: SR1.AlyzrReferences.dBmZ(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.dBmZ("")
SR1.AlyzrReferences.dBmZ("") = 600.0
```

Description: Reference impedance used in dBm calculations.

WattsZ

Command Syntax: SR1.AlyzrReferences.WattsZ(*ValueUnit*)

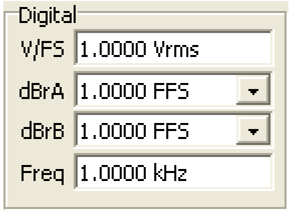
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.WattsZ("")
SR1.AlyzrReferences.WattsZ("") = 8.0
```

Description: Terminating impedance used for calculating Watts.



Digital

V/FS 1.0000 Vrms

dBrA 1.0000 FFS

dBrB 1.0000 FFS

Freq 1.0000 kHz

DigdBrA

Command Syntax: SR1.AlyzrReferences.DigdBrA(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.DigdBrA("Vrms")
SR1.AlyzrReferences.DigdBrA("Vrms") = 1.0
```

Description: Reference value for digital audio dBrA units.

DigdBrB

Command Syntax: SR1.AlyzrReferences.DigdBrB(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.DigdBrB("Vrms")
SR1.AlyzrReferences.DigdBrB("Vrms") = 1.0
```

Description: Reference value for digital audio dBrB units.

DigFreq

Command Syntax: SR1.AlyzrReferences.DigFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AlyzrReferences.DigFreq("Hz")
SR1.AlyzrReferences.DigFreq("Hz") = 1000.0
```

Description: Digital audio reference frequency.

DigVfs

Command Syntax: SR1.AlyzrReferences.DigVfs(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

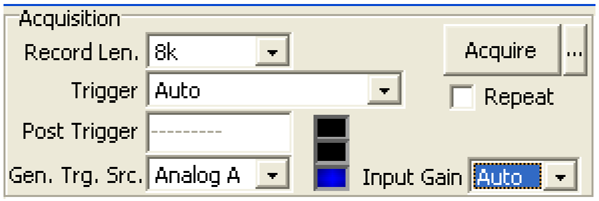
Example:

```
val = SR1.AlyzrReferences.DigVfs("Vrms")
SR1.AlyzrReferences.DigVfs("Vrms") = 1.0
```

Description: Digital audio "Volts Full Scale" value.

2.4.9 Digitizer

Object:	**SR1.Digitizer**
Object Argument(s):	None
Description:	Commands related to the optional digital audio carrier digitizer.



Record Len. 8k

Trigger Auto

Post Trigger ---------

Gen. Trg. Src. Analog A

□ Repeat

Input Gain Auto

RecordLength

Command Syntax: SR1.Digitizer.RecordLength

Command Argument(s):

Property: dgtzrRecLen enumeration {drl4k=0 | drl8k=1 | drl16k=2 | drl32k=3 | drl64k=4 | drl128k=5 | drl256k=6 | drl512k=7 | drl1M=8 | drl2M=9}

Example:

```
val = SR1.Digitizer.RecordLength
SR1.Digitizer.RecordLength = drl4k
```

Description: Digitizer record length.

Repeat

Command Syntax: SR1.Digitizer.Repeat

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Digitizer.Repeat
SR1.Digitizer.Repeat = False
```

Description: Digitizer "repeat acquisition" status.

InputStatusRdg

Command Syntax: SR1.Digitizer.InputStatusRdg

Command Argument(s):

Property: DgtzrInStat enumeration (read-only) {dgUnderRange=-1 | dgInRange=0 | dgOverRange=1}

Example:

```
val = SR1.Digitizer.InputStatusRdg
```

Description: Digitizer overload status.

TriggerSource

Command Syntax: SR1.Digitizer.TriggerSource

Command Argument(s):

Property: DigiTrig enumeration {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:

```
val = SR1.Digitizer.TriggerSource
SR1.Digitizer.TriggerSource = Auto
```

Description: Digitizer trigger source.

PostTrig

Command Syntax: SR1.Digitizer.PostTrig(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Digitizer.PostTrig("%")
SR1.Digitizer.PostTrig("%") = 50
```

Description: Percentage of the digitizer record that will follow the trigger.

Acquire

Command Syntax: SR1.Digitizer.Acquire(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.Digitizer.Acquire("")
```

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: SR1.Digitizer.Cancel()

Command Argument(s): None

Example:

```
Call SR1.Digitizer.Cancel()
```

Description: Cancels any digitizer acquisition in progress.

Save

Command Syntax: SR1.Digitizer.Save(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.Digitizer.Save("DigFile.txt")
```

Description: Saves the current digitizer input data to the name file.

State

Command Syntax: SR1.Digitizer.State()

Command Argument(s): None

Return Value: *State* as dgtzrState enumeration {dsIdle=0 | dsWaitingForTrig=1 | dsAcquiring=2 | dsAnalyzing=3 | dsCancelling=4}

Example: `State = SR1.Digitizer.State()`

Description: Returns the current state of the digitizer.

InputGain

Command Syntax: SR1.Digitizer.InputGain

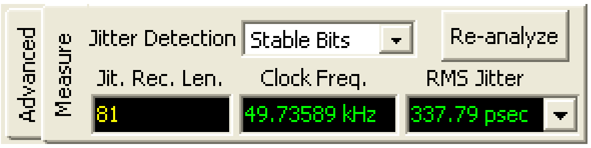
Command Argument(s):

Property: DgtzrGain enumeration {dgAuto=0 | dgx1=1 | dgx2=2 | dgx4=3}

Example:

```
val = SR1.Digitizer.InputGain
SR1.Digitizer.InputGain = dgAuto
```

Description: Digitizer input gain setting.



Jitter Detection: Stable Bits

Jit. Rec. Len.: 81

Clock Freq.: 49.73589 kHz

RMS Jitter: 337.79 psec

JitterDetection

Command Syntax: SR1.Digitizer.JitterDetection

Command Argument(s):

Property: JitterDet enumeration {AllBits=0 | StableBits=1 | Preamble=2 | SqRising=3 | SqFalling=4 | SqBoth=5}

Example:

```
val = SR1.Digitizer.JitterDetection
SR1.Digitizer.JitterDetection = AllBits
```

Description: Digitizer transition selection. AllBits means transitions every 2 UI's are analyzed, StableBits means transitions every 8 UI's are analyzed, and Preamble means transitions every 64 UI's are analyzed. For square wave input, the options are rising edges only, falling edges only, or both rising and falling edges.

Reanalyze

Command Syntax: SR1.Digitizer.Reanalyze()

Command Argument(s): None

Example: `Call SR1.Digitizer.Reanalyze()`

Description: Reanalyzes the current digitizer data without acquiring new data.

JitterRecLenRdg

Command Syntax: SR1.Digitizer.JitterRecLenRdg

Command Argument(s):

Property: Integer (read-only)

Example: `val = SR1.Digitizer.JitterRecLenRdg`

Description: Total number of jitter points calculated given the digitizer record length and the stable bits selection.

ClockFreqRdg

Command Syntax: SR1.Digitizer.ClockFreqRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Digitizer.ClockFreqRdg("Hz")`

Description: Calculated effective digital audio sampling clock calculated from the digitizer data.

RMSJitterRdg

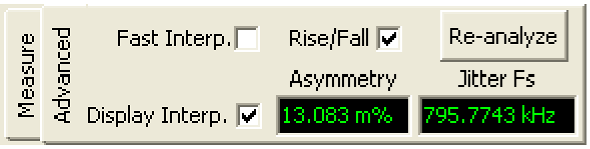
Command Syntax: SR1.Digitizer.RMSJitterRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Digitizer.RMSJitterRdg("sec")`

Description: Calculated total jitter from the digitizer data.



Fast Interp. □

Rise/Fall ☑

Display Interp. ☑

Asymmetry 13.083 m%

Jitter Fs 795.7743 kHz

FastInterpolation

Command Syntax: SR1.Digitizer.FastInterpolation

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Digitizer.FastInterpolation
SR1.Digitizer.FastInterpolation = False
```

Description: On/off status of digitizer Fast Interpolation. Turning on fast interpolation (spline) enables a faster but less accurate jitter calculation algorithm.

RiseFallEdges

Command Syntax: SR1.Digitizer.RiseFallEdges

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Digitizer.RiseFallEdges
SR1.Digitizer.RiseFallEdges = False
```

Description: On/off state of the asymmetry correction feature of the digitizer. If on, it removes jitter contribution due to DC offsets and rise/fall time asymmetry.

DisplayInterp

Command Syntax: SR1.Digitizer.DisplayInterp

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Digitizer.DisplayInterp
SR1.Digitizer.DisplayInterp = False
```

Description: .On/off status of digitizer display interpolation.

AsymmetryRdg

Command Syntax: SR1.Digitizer.AsymmetryRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Digitizer.AsymmetryRdg("")
```

Description: Asymmetry value calculated by the digitizer. 0% means perfect symmetry, while 100% means highly asymmetric rise/fall times.

JitterFsRdg

Command Syntax: SR1.Digitizer.JitterFsRdg(*ValueUnit*)

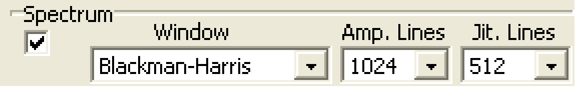
Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Digitizer.JitterFsRdg("")
```

Description: Effective jitter sampling rate determined by the input digital audio sampling rate and the jitter detection selection.



Spectrum ☑

Window: Blackman-Harris

Amp. Lines: 1024

Jit. Lines: 512

CalcSpectrum

Command Syntax: SR1.Digitizer.CalcSpectrum

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Digitizer.CalcSpectrum
SR1.Digitizer.CalcSpectrum = False
```

Description: On/off status of digitizer spectrum measurement calculation.

SpectrumWindow

Command Syntax: SR1.Digitizer.SpectrumWindow

Command Argument(s):

Property: DigitizerWindowTypes enumeration {BlkmanHarris=0 | Hann=1 | Flattop=2 | EqRipple=3 | Uniform=4}

Example:

```
val = SR1.Digitizer.SpectrumWindow
SR1.Digitizer.SpectrumWindow = BlkmanHarris
```

Description: Window selection for digitizer spectrum measurements.

InputSpecLines

Command Syntax: SR1.Digitizer.InputSpecLines

Command Argument(s):

Property: InSpecLines enumeration {il512=0 | il1k=1 | il2k=2 | il4k=3 | il8k=4 | il16k=5}

Example:

```
val = SR1.Digitizer.InputSpecLines
SR1.Digitizer.InputSpecLines = il512
```

Description: Number of lines of resolution for the input amplitude spectra.

JitterSpecLines

Command Syntax: SR1.Digitizer.JitterSpecLines

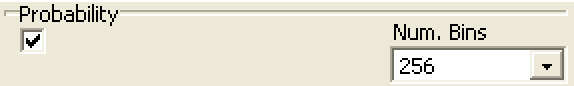
Command Argument(s):

Property: JitSpecLines enumeration {jl256=0 | jl512=1 | jl1k=2 | jl2k=3 | jl4k=4 | jl8k=5}

Example:

```
val = SR1.Digitizer.JitterSpecLines
SR1.Digitizer.JitterSpecLines = jl256
```

Description: Number of lines of resolution for the jitter spectra



Probability

☑

Num. Bins

256

CalcProbability

Command Syntax: SR1.Digitizer.CalcProbability

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Digitizer.CalcProbability
SR1.Digitizer.CalcProbability = False
```

Description: On/off status of digitizer probability measurement calculation..

NumBins

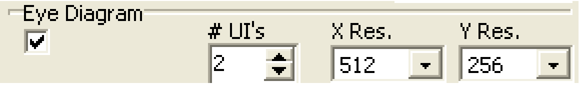
Command Syntax: SR1.Digitizer.NumBins

Command Argument(s):

Property: EyeBins enumeration {eb64=0 | eb128=1 | eb256=2 | eb512=3 | eb1024=4}

Example:
```
val = SR1.Digitizer.NumBins
SR1.Digitizer.NumBins = eb64
```

Description: Number of histogram bins for digitizer probability measurements.



Eye Diagram ☑

UI's: 2

X Res.: 512

Y Res.: 256

CalcEyeDiag

Command Syntax: SR1.Digitizer.CalcEyeDiag

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Digitizer.CalcEyeDiag
SR1.Digitizer.CalcEyeDiag = False
```

Description: On/off status of eye-diagram calculations.

NumEyes

Command Syntax: SR1.Digitizer.NumEyes

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Digitizer.NumEyes
SR1.Digitizer.NumEyes = 2
```

Description: Number of eyes (unit intervals) in the eye-diagram.

EyeXRez

Command Syntax: SR1.Digitizer.EyeXRez

Command Argument(s):

Property: EyeRez enumeration {er128=0 | er256=1 | er512=2 | er1024=3}

Example:

```
val = SR1.Digitizer.EyeXRez
SR1.Digitizer.EyeXRez = er128
```

Description: X-resolution of the eye-diagram.

EyeYRez

Command Syntax: SR1.Digitizer.EyeYRez

Command Argument(s):

Property: EyeRez enumeration {er128=0 | er256=1 | er512=2 | er1024=3}

Example:

```
val = SR1.Digitizer.EyeYRez
SR1.Digitizer.EyeYRez = er128
```

Description: Y-resolution of the eye-diagram.

GetEyeRez

Command Syntax: SR1.Digitizer.GetEyeRez()

Command Argument(s): None

Return Value: *NumXY* as Array

Example:

```
NumXY = SR1.Digitizer.GetEyeRez()
```

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: SR1.Digitizer.GetXData(*DgtzrMeas*, *UnitString*)

Command Argument(s): *DgtzrMeas* as DgtzrMeasID enumeration {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 as String

Return Value: *Data* as Array

Example:

```
Data = SR1.Digitizer.GetXData(dmInputTimeRec, "s")
```

Description: X-axis data array for the specified measurement.

GetYData

Command Syntax: SR1.Digitizer.GetYData(*DgtzrMeas*, *UnitString*)

Command Argument(s): *DgtzrMeas* as DgtzrMeasID enumeration {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 as String

Return Value: *Data* as Array

Example: `Data = SR1.Digitizer.GetYData(dmInputTimeRec, "Vp")`

Description: Y-axis data array for the specified measurement.

GetZData

Command Syntax: SR1.Digitizer.GetZData(*DgtzrMeas*, *UnitString*)

Command Argument(s): *DgtzrMeas* as DgtzrMeasID enumeration {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 as String

Return Value: *Data* as Array

Example: `Data = SR1.Digitizer.GetZData(dmEyeIntensity, "")`

Related Command(s): :Digitizer:GetEyeRez?

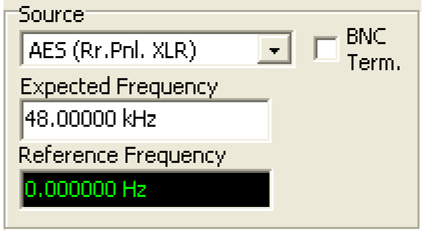
Description: Eye diagram intensity data. (Currently, the only supported measurement argument for this command is "8" (dmEyeIntensity)) The intensity data is returned in a "raster-scan" fashion starting from the top left of the eye diagram and ending with the bottom right. The number of cells in the eye diagram can be determined using the GetEyeRez? command.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.10 Clock References

Object:	**SR1.ClockRef**
Object Argument(s):	None
Description:	Commands related to the Clock Reference panel.



Source

AES (Rr.Pnl. XLR)

□ BNC Term.

Expected Frequency

48.00000 kHz

Reference Frequency

0.000000 Hz

Source

Command Syntax: SR1.ClockRef.Source

Command Argument(s):

Property: CRSource enumeration {crAES=0 | crExt=1 | crInt=2 | crNTSC=3 | crSECAM=4 | crPAL=5}

Example:

```
val = SR1.ClockRef.Source
SR1.ClockRef.Source = crAES
```

Description: Sets or gets the clock source.

ExtTerm

Command Syntax: SR1.ClockRef.ExtTerm

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.ClockRef.ExtTerm
SR1.ClockRef.ExtTerm = False
```

Description: Termination status of the rear-panel BNC clock input.

AESFreq

Command Syntax: SR1.ClockRef.AESFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.ClockRef.AESFreq("Hz")
SR1.ClockRef.AESFreq("Hz") = 48000.0
```

Description: Expected frequency of the AES reference input.

ExtFreq

Command Syntax: SR1.ClockRef.ExtFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.ClockRef.ExtFreq("Hz")
SR1.ClockRef.ExtFreq("Hz") = 10000000
```

Description: Expected frequency for the rear panel external reference BNC input.

NTSCFreq

Command Syntax: SR1.ClockRef.NTSCFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.ClockRef.NTSCFreq("Hz")
SR1.ClockRef.NTSCFreq("Hz") = 15734.3
```

Description: Expected frequency for the rear panel video input when an NTSC video source is used.

PALFreq

Command Syntax: SR1.ClockRef.PALFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.ClockRef.PALFreq("Hz")
SR1.ClockRef.PALFreq("Hz") = 15625.0
```

Description: Expected frequency for the rear panel video input when a PAL video source is used.

SECAMFreq

Command Syntax: SR1.ClockRef.SECAMFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.ClockRef.SECAMFreq("Hz")
SR1.ClockRef.SECAMFreq("Hz") = 15625.0
```

Description: Expected frequency for the rear panel video input when a SECAM video source is used.

FreqRdg

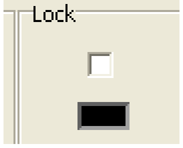
Command Syntax: SR1.ClockRef.FreqRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.ClockRef.FreqRdg("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: SR1.ClockRef.DoLock

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.ClockRef.DoLock
SR1.ClockRef.DoLock = False
```

Description: 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.

LockStatus

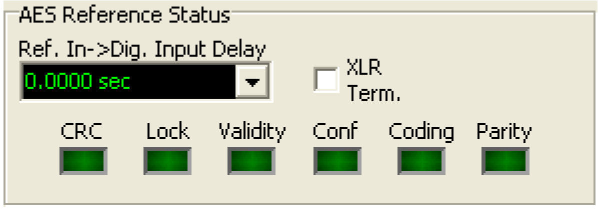
Command Syntax: SR1.ClockRef.LockStatus

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example: `val = SR1.ClockRef.LockStatus`

Description: Returns true if SR1 is currently locked to an external source. Returns false if SR1 cannot lock to the specified source.


AES Reference Status

Ref. In->Dig. Input Delay: 0.0000 sec

□ XLR Term.

RefInDelayRdg

Command Syntax: SR1.ClockRef.RefInDelayRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.ClockRef.RefInDelayRdg("s")
```

Description: Returns the delay between the rear panel AES reference input and the front-panel digital audio input.

AESTerm

Command Syntax: SR1.ClockRef.AESTerm

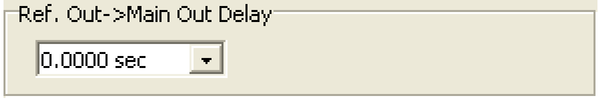
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.ClockRef.AESTerm
SR1.ClockRef.AESTerm = False
```

Description: Termination on/off status for the rear-panel AES reference input.



Ref. Out->Main Out Delay

0.0000 sec

DelayOutFromRef

Command Syntax: SR1.ClockRef.DelayOutFromRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.ClockRef.DelayOutFromRef("UI")
SR1.ClockRef.DelayOutFromRef("UI") = 127.500
```

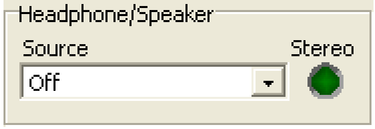
Description: Delay setting between the AES reference output and the main digital audio. The delay is adjustable between 0 and 128 UI (one full digital audio frame).

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.11 Monitors

Object:	**SR1.Monitor**
Object Argument(s):	None
Description:	Commands related to the Monitors panel.



Source: Off

Source

Command Syntax: SR1.Monitor.Source

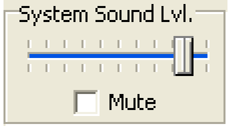
Command Argument(s):

Property: SoundSrcs enumeration {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:

```
val = SR1.Monitor.Source
SR1.Monitor.Source = ssOff
```

Description: Source for the Speaker/Headphones.



□ Mute

SysMute

Command Syntax: SR1.Monitor.SysMute

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Monitor.SysMute
SR1.Monitor.SysMute = False
```

Description: Mute status for Windows generated sounds.

SysSound

Command Syntax: SR1.Monitor.SysSound

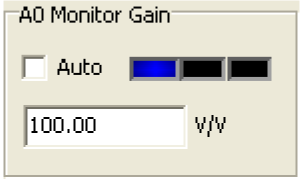
Command Argument(s):

Property: Integer

Example:

```
val = SR1.Monitor.SysSound
SR1.Monitor.SysSound = Value
```

Description: Windows sound level. 0 to 100.



A0 Monitor Gain

□ Auto

100.00 V/V

A0AutoGain

Command Syntax: SR1.Monitor.A0AutoGain

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Monitor.A0AutoGain
SR1.Monitor.A0AutoGain = False
```

Description: On/off status of A0 monitor autoranging.

A0Gain

Command Syntax: SR1.Monitor.A0Gain(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Monitor.A0Gain("")
SR1.Monitor.A0Gain("") = 1.0000
```

Description: A0 monitor gain.

A0Status

Command Syntax: SR1.Monitor.A0Status

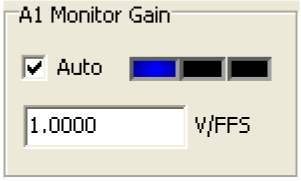
Command Argument(s):

Property: MonitorStatus enumeration (read-only) {msUnderRange=0 | msInRange=1 | msOverRange=2}

Example:

```
val = SR1.Monitor.A0Status
```

Description: Overload status for the A0 monitor.


A1 Monitor Gain

☑ Auto

1.0000 V/FFS

A1AutoGain

Command Syntax: SR1.Monitor.A1AutoGain

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Monitor.A1AutoGain
SR1.Monitor.A1AutoGain = False
```

Description: On/off status of A1 monitor autoranging.

A1Gain

Command Syntax: SR1.Monitor.A1Gain(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Monitor.A1Gain("")
SR1.Monitor.A1Gain("") = 10.0
```

Description: A1 Monitor gain.

A1Status

Command Syntax: SR1.Monitor.A1Status

Command Argument(s):

Property: MonitorStatus enumeration (read-only) {msUnderRange=0 | msInRange=1 | msOverRange=2}

Example:
```
val = SR1.Monitor.A1Status
```

Description: A1 monitor overload status.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount

FormID

2.4.12 Instrument

Object:	**SR1.Instrument**
Object Argument(s):	None
Description:	Commands related to the instrument user interface.

Form and Page Control Commands

ActivePage

Command Syntax: SR1.Instrument.ActivePage

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Instrument.ActivePage
SR1.Instrument.ActivePage = 2
```

Description: Active page (1-7) of the main page control.

CloseForm

Command Syntax: SR1.Instrument.CloseForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example:
```
Call SR1.Instrument.CloseForm(Value)
```

Description: Closes a form of any type, on any page of the page control, with the specified FormID.

FormCount

Command Syntax: SR1.Instrument.FormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example:
```
Count = SR1.Instrument.FormCount()
```

Description: Returns the number of forms on the current page of the page control.

FormID

Command Syntax: SR1.Instrument.FormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example:
```
FormID = SR1.Instrument.FormID(Value)
```

Description: Returns the FormID of the Indexth form on the current page of the page control.

GetFormPage

Command Syntax: SR1.Instrument.GetFormPage(*FormID*)

Command Argument(s): *FormID* as Integer

Return Value: *Page* as Integer

Example: `Page = SR1.Instrument.GetFormPage(101)`

Description: Returns the page of the page control (1-7) containing the form with the given page control.

GetFormPos

Command Syntax: SR1.Instrument.GetFormPos(*FormID*)

Command Argument(s): *FormID* as Integer

Return Value: *Pos* as Array

Example: `Pos = SR1.Instrument.GetFormPos(5)`

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: SR1.Instrument.GetFormSize(*FormID*)

Command Argument(s): *FormID* as Integer

Return Value: *Size* as Array

Example: `Size = SR1.Instrument.GetFormSize(5)`

Description: Returns the size in pixels (X,Y) of the specified form.

GetFormState

Command Syntax: SR1.Instrument.GetFormState(*FormID*)

Command Argument(s): *FormID* as Integer

Return Value: *State* as FormState enumeration {fsNormal=0 | fsMinimized=1 | fsMaximized=2}

Example: `State = SR1.Instrument.GetFormState(Value)`

Description: Returns the state of the form with the given FormID.

GetFormTitle

Command Syntax: SR1.Instrument.GetFormTitle(*FormID*)

Command Argument(s): *FormID* as Integer

Return Value: *Title* as String

Example: `Title = SR1.Instrument.GetFormTitle(5)`

Description: Returns the title string of the specified form.

GetMainFormSize

Command Syntax: SR1.Instrument.GetMainFormSize()

Command Argument(s): None

Return Value: *Size* as Array

Example: `Size = SR1.Instrument.GetMainFormSize()`

Description: Returns the size (X,Y), in pixels, of the main form.

MoveForm

Command Syntax: SR1.Instrument.MoveForm(*FormID*, *Page*)

Command Argument(s): *FormID* as Integer
Page as Integer

Example: `Call SR1.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: SR1.Instrument.MoveSelectedForm(*Page*)

Command Argument(s): *Page* as Integer

Example: `Call SR1.Instrument.MoveSelectedForm(3)`

Description: Moves the currently selected form to the specified page (1-7) on the page control.

SelectedForm

Command Syntax: SR1.Instrument.SelectedForm()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Instrument.SelectedForm()`

Description: Returns the FormID of the currently selected form.

SelectForm

Command Syntax: SR1.Instrument.SelectForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.Instrument.SelectForm(5)`

Description: Selects the form with the specified FormID.

TileForms

Command Syntax: SR1.Instrument.TileForms()

Command Argument(s): None

Example: `Call SR1.Instrument.TileForms()`

Description: Tiles the forms on the current page of the page control.

SetFormPos

Command Syntax: SR1.Instrument.SetFormPos(*FormID*, *Pos*)

Command Argument(s): *FormID* as Integer
Pos as Array

Example:
```
Call SR1.Instrument.SetFormPos(20, Pos)
```

Description: Sets the position of the form with the given FormID to the X-Y position specified in the array Pos.

SetFormSize

Command Syntax: SR1.Instrument.SetFormSize(*FormID*, *Size*)

Command Argument(s): *FormID* as Integer
Size as Array

Example:
```
Call SR1.Instrument.SetFormSize(25, Size)
```

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: SR1.Instrument.SetFormState(*FormID*, *State*)

Command Argument(s): *FormID* as Integer
State as FormState enumeration {fsNormal=0 | fsMinimized=1 | fsMaximized=2}

Example:
```
Call SR1.Instrument.SetFormState(Value, fsNormal)
```

Description: Sets the state (normal, minimized, maximized) of the form with the given FormID.

Measurement Commands

LastMeas

Command Syntax: SR1.Instrument.LastMeas(*MeasID*, *UnitStr*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as String

Return Value: *Meas* as Double

Example: `Meas = SR1.Instrument.LastMeas(msAnlgFreqA, "Hz")`

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: SR1.Instrument.LastVectorBinMeas(*MeasID*, *BinIndex*, *UnitStr*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as Integer
UnitStr as String

Return Value: *Meas* as Double

Example: `Meas = SR1.Instrument.LastVectorBinMeas(msA1FFTspectrum,`

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: SR1.Instrument.LastVectorBinXMeas(*MeasID*, *BinIndex*, *UnitStr*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as Integer
UnitStr as String

Return Value: *Meas* as Double

Example: `Meas = SR1.Instrument.LastVectorBinXMeas(msA1FFTspectrum`

Description: X-axis value corresponding the the specified bin of the specified measurement.

LastVectorMeas

Command Syntax: SR1.Instrument.LastVectorMeas(*MeasID*, *UnitStr*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as String

Return Value: *Vector* as Array

Example: `Vector = SR1.Instrument.LastVectorMeas(msA1FFTtimeRec, "`

Description: Returns an entire array corresponding to the last value of a vector measurement.

LastVectorXMeas

Command Syntax: SR1.Instrument.LastVectorXMeas(*MeasID*, *UnitStr*)

Command Argument(s): *MeasID* as MeasID enumeration {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 |
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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as String

Return Value: *Vector* as Array

Example: `Vector = SR1.Instrument.LastVectorXMeas(msA1FFTspectrum,`

Description: Returns an entire array corresponding to the X-axis of a vector measurement.

MeasCount

Command Syntax: SR1.Instrument.MeasCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Instrument.MeasCount()`

Description: Number of active measurements currently being made by SR1.

MeasInfo

Command Syntax: SR1.Instrument.MeasInfo(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *Info* as String

Example: `Info = SR1.Instrument.MeasInfo(msA1FFTspectrum)`

Description: Returns a string describing the specified measurement.

MeasItem

Command Syntax: SR1.Instrument.MeasItem(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.Instrument.MeasItem(10)`

Description: Returns the measurement ID for ith 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: SR1.Instrument.MeasUnit(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *UnitStr* as String

Example: `UnitStr = SR1.Instrument.MeasUnit(msA1FFTspectrum)`

Description: Returns a string corresponding to the default units for measurement with the specified MeasID.

XUnitStr

Command Syntax: SR1.Instrument.XUnitStr(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *UnitStr* as String

Example: `UnitStr = SR1.Instrument.XUnitStr(msA1FFTspectrum)`

Description: Returns a string corresponding to the default units for the x-axis of vector measurement with the specified MeasID.

SettlerIndex

Command Syntax: SR1.Instrument.SettlerIndex(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 |

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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *SettlerIndex* as Integer

Example: `SettlerIndex = SR1.Instrument.SettlerIndex(msA0HistoProb`

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 SR1.Sweep.Settling(i).

MeasGetStorageDepth

Command Syntax: SR1.Instrument.MeasGetStorageDepth(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *Depth* as Integer

Example: `Depth = SR1.Instrument.MeasGetStorageDepth(msAnlgFreqA)`

Description: Maximum storage depth for the measurement specified.

MeasSetStorageDepth

Command Syntax: SR1.Instrument.MeasSetStorageDepth(*MeasID*, *Depth*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}
Depth as Integer

Example: `Call SR1.Instrument.MeasSetStorageDepth(msA1FFTspectrum,`

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: SR1.Instrument.MeasGetNumStoredData(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *N* as Integer

Example: `N = SR1.Instrument.MeasGetNumStoredData(msNull)`

Description: Current storage depth for the specified measurement.

StoredVectorMeas

Command Syntax: SR1.Instrument.StoredVectorMeas(*MeasID*, *UnitStr*, *Index*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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 as String
Index as Integer

Return Value: *Vector* as Array

Example: `Vector = SR1.Instrument.StoredVectorMeas(msA1FFTspectrum`

Description: Returns the vector corresponding to the Indexth stored measurement of type MeasID. Index = 0 corresponds to the oldest measurement, increasing index corresponds to more recent measurements.

User Input Commands

UserChoice

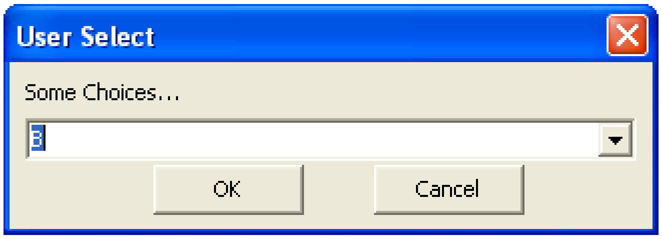
Command Syntax: SR1.Instrument.UserChoice(*Message*, *Choices*, *Timeout*)

Command Argument(s): *Message* as String
Choices as String
Timeout as Integer

Return Value: *Result* as String

Example: `Result = SR1.Instrument.UserChoice("Some Choices...", "A`

Description: Displays a drop-down user selection window:



Some Choices...

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

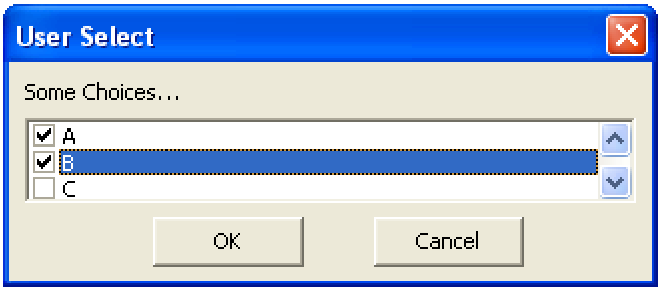
Command Syntax: SR1.Instrument.UserChoiceMulti(*Message*, *Choices*, *Timeout*)

Command Argument(s): *Message* as String
Choices as String
Timeout as Integer

Return Value: *Result* as String

Example: `Result = SR1.Instrument.UserChoiceMulti("Some Choices...`

Description: Displays a checkbox user selection window:



User Select

Some Choices...

☑ A
☑ B
□ C

The "Message" string is displayed over a 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

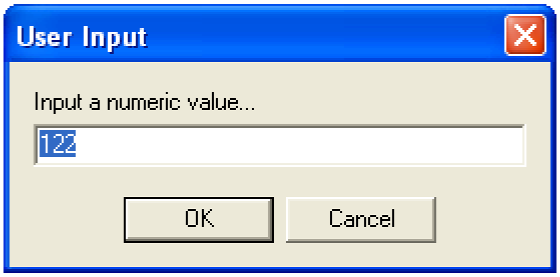
Command Syntax: SR1.Instrument.UserInput(*Message*, *DefaultResult*, *Timeout*)

Command Argument(s): *Message* as String
DefaultResult as String
Timeout as Integer

Return Value: *Result* as String

Example: `Result = SR1.Instrument.UserInput("Input a numeric value`

Description: Displays a window with a single edit control for user input:



Input a numeric value...

122

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: SR1.Instrument.UserLaunchChoice(*Message*, *Buttons*, *Descriptions*, *Timeout*)

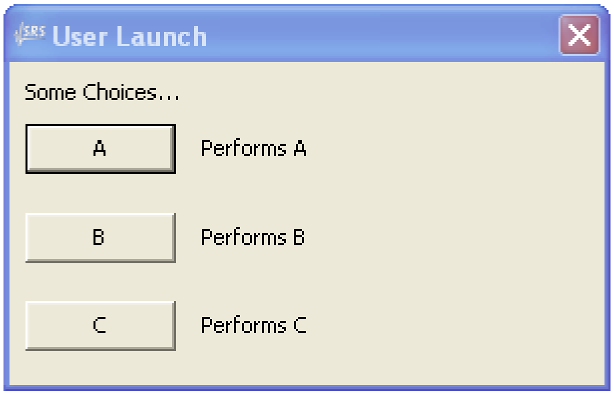
Command Argument(s): *Message* as String
Buttons as String
Descriptions as String
Timeout as Integer

Return Value: *Result* as String

Example:

```
Result = SR1.Instrument.UserLaunchChoice("Some Choices..
```

Description: Displays a window with a series of labeled buttons:



Some Choices...

Performs A

Performs B

Performs C

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 tot 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

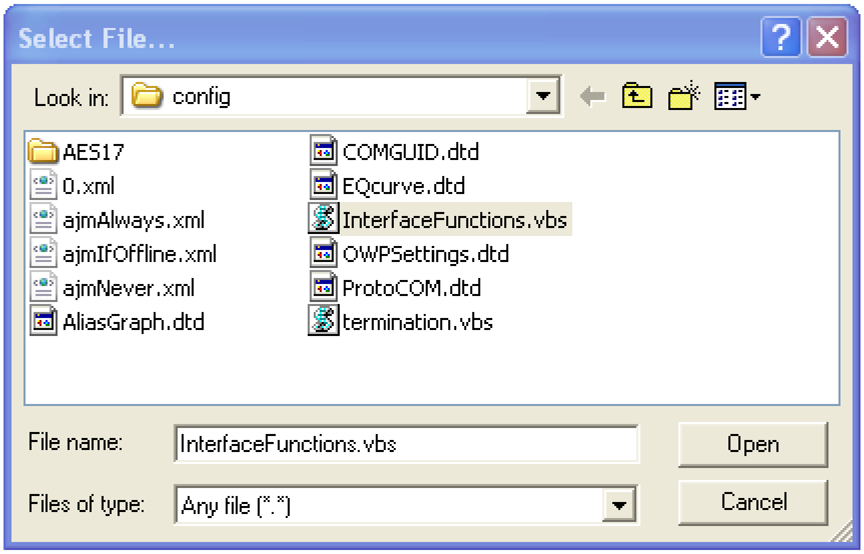
Command Syntax: SR1.Instrument.UserLoadFile(*Message*, *Timeout*)

Command Argument(s): *Message* as String
Timeout as Integer

Return Value: *Result* as String

Example: `Result = SR1.Instrument.UserLoadFile("Select File...", 1`

Description: Displays a file load dialog box:



Select File...

Look in: config

File name: InterfaceFunctions.vbs

Files of type: Any file (*.*)

The query returns the string containing the path and filename of the selected file, or "-cancelled-".

UserMessage

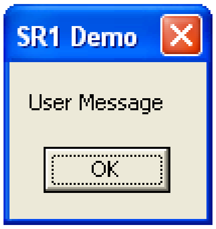
Command Syntax: SR1.Instrument.UserMessage(*Message*, *Timeout*)

Command Argument(s): *Message* as String
Timeout as Integer

Return Value: *Result* as UserMessageResult enumeration {umTimedout=-1 | umOK=1}

Example: `Result = SR1.Instrument.UserMessage("User Message", 100)`

Description: Displays a window with a user message:



SR1 Demo

User Message

The query returns either umTimedout or umOK depending on whether the user presses "OK" before the timeout interval expires.

UserOKCancel

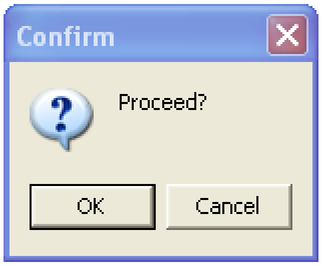
Command Syntax: SR1.Instrument.UserOKCancel(*Message*, *Timeout*)

Command Argument(s): *Message* as String
Timeout as Integer

Return Value: *Result* as OKCancelResult enumeration {ocTimedout=-1 | ocCancel=0 | ocOK=1}

Example: `Result = SR1.Instrument.UserOKCancel("Value", Value)`

Description: Displays a window with a user message and an OK and Cancel button.


Confirm

Proceed?

The query returns ocTimedout, ocCancel, or ocOK depending on the user action.

UserSaveFile

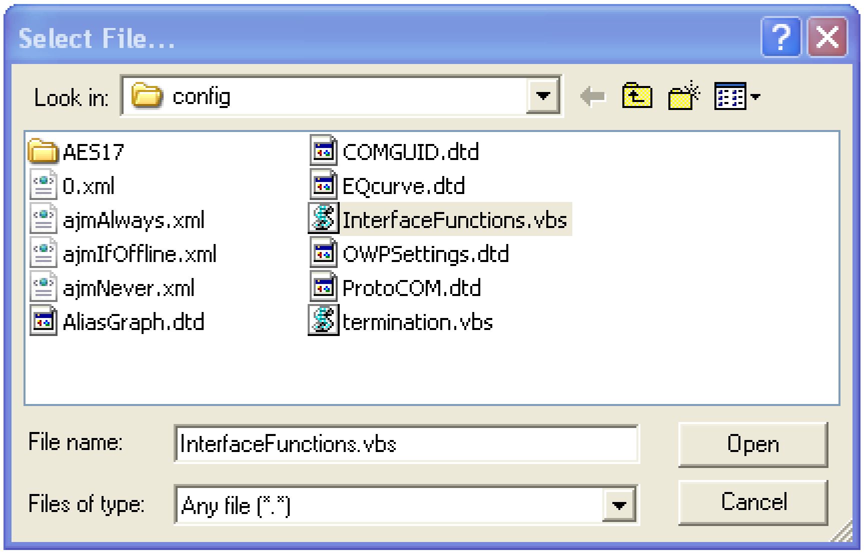
Command Syntax: SR1.Instrument.UserSaveFile(*Message*, *Timeout*)

Command Argument(s): *Message* as String
Timeout as Integer

Return Value: *Result* as String

Example: `Result = SR1.Instrument.UserSaveFile("Value", Value)`

Description: Displays a file save dialog box:


Select File...

Look in: config

AES17
0.xml
ajmAlways.xml
ajmIfOffline.xml
ajmNever.xml
AliasGraph.dtd
COMGUID.dtd
EQcurve.dtd
InterfaceFunctions.vbs
OWPSettings.dtd
ProtoCOM.dtd
termination.vbs

File name: InterfaceFunctions.vbs

Files of type: Any file (*.*)

The query returns the string containing the path and filename of the selected file, or "-cancelled-".

UserYesNoCancel

Command Syntax: SR1.Instrument.UserYesNoCancel(*Message*, *Timeout*)

Command Argument(s): *Message* as String
Timeout as Integer

Return Value: *Result* as YesNoCancelResult enumeration {yncTimedout=-1 | yncNo=0 | yncYes=1 | yncCancel=2}

Example:
```
Result = SR1.Instrument.UserYesNoCancel("Value", Value)
```

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: SR1.Instrument.Wait(*Duration*)

Command Argument(s): *Duration* as Integer

Example:
```
Call SR1.Instrument.Wait(1000)
```

Description: Causes a delay for *Duration* milliseconds

Misc. Instrument Commands

Load

Command Syntax: SR1.Instrument.Load(*FileName*)

Command Argument(s): *FileName* as String

Example:
```
Call SR1.Instrument.Load("MyConfig.xml")
```

Description: Loads the specified SR1 configuration file.

LoadPartial

Command Syntax: SR1.Instrument.LoadPartial(*FileName*, *Mask*)

Command Argument(s): *FileName* as String
Mask as FileMaskType enumeration {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:
```
Call SR1.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: SR1.Instrument.Beep()

Command Argument(s): None

Example: `Call SR1.Instrument.Beep()`

Description: Makes a "beep" sound.

Reset

Command Syntax: SR1.Instrument.Reset()

Command Argument(s): None

Example: `Call SR1.Instrument.Reset()`

Description: Resets the instrument to its default state.

Save

Command Syntax: SR1.Instrument.Save(*FileName*, *GraphData*)

Command Argument(s): *FileName* as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Instrument.Save("MyConfig.xml", gdNeverSave)`

Description: Saves the entire instrument configuration. The GraphData argument specifies whether to include graph data in the configuration file.

SavePartial

Command Syntax: SR1.Instrument.SavePartial(*FileName*, *Mask*, *GraphData*)

Command Argument(s): *FileName* as String
Mask as FileMaskType enumeration {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}
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Instrument.SavePartial("Value", fmAll, gdNeverS`

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: SR1.Instrument.SerialNo()

Command Argument(s): None

Return Value: *SN* as String

Example: `SN = SR1.Instrument.SerialNo()`

Description: Returns the serial number of the instrument.

Version

Command Syntax: SR1.Instrument.Version()

Command Argument(s): None

Return Value: *Ver* as String

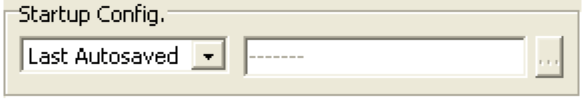
Example: `Ver = SR1.Instrument.Version()`

Description: Returns the version string corresponding to the instrument software.

2.4.13 Preferences

Object:	**SR1.Preferences**
Object Argument(s):	None
Description:	Commands related to the Preferences panel.

General Preference Commands



Startup Config.

Last Autosaved

StartupConfig

Command Syntax: SR1.Preferences.StartupConfig

Command Argument(s):

Property: StartupChoice enumeration {scSR1Default=0 | scUserDefault=1 | scLastSaved=2}

Example:

```
val = SR1.Preferences.StartupConfig
SR1.Preferences.StartupConfig = scSR1Default
```

Description: Startup SR1 configuration mode.

UserDefaultConfigFile

Command Syntax: SR1.Preferences.UserDefaultConfigFile

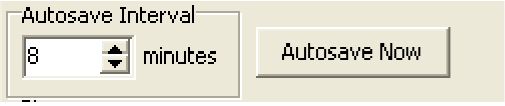
Command Argument(s):

Property: String

Example:

```
val = SR1.Preferences.UserDefaultConfigFile
SR1.Preferences.UserDefaultConfigFile = "MyStartup.XML"
```

Description: Startup configuration file used in "User Default" mode.



Autosave Interval

8 minutes

AutosaveInterval

Command Syntax: SR1.Preferences.AutosaveInterval

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Preferences.AutosaveInterval
SR1.Preferences.AutosaveInterval = Value
```

Description: time interval between autosaves.

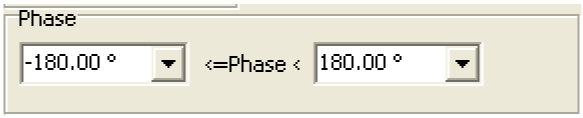
AutosaveNow

Command Syntax: SR1.Preferences.AutosaveNow(*SaveDuringSweep*)

Command Argument(s): *SaveDuringSweep* as Boolean {False=0 | True=1}

Example:
```
Call SR1.Preferences.AutosaveNow(False)
```

Description: Saves the current instrument configuration to the autosave file.



Phase

-180.00 ° <=Phase < 180.00 °

PhaseMax

Command Syntax: SR1.Preferences.PhaseMax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Preferences.PhaseMax("deg")
SR1.Preferences.PhaseMax("deg") = 180
```

Description: Maximum value of the phase measurement range.

PhaseMin

Command Syntax: SR1.Preferences.PhaseMin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Preferences.PhaseMin("")
SR1.Preferences.PhaseMin("") = Value
```

Description: Minimum value of the phase measurement range.

Generator Signal Initialization

Last Used

SignalInit

Command Syntax: SR1.Preferences.SignalInit

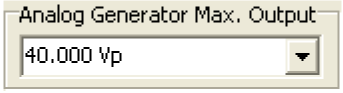
Command Argument(s):

Property: SignalInitialization enumeration {siSR1Default=0 | siLastUsed=1 | siLastUsedExclAmp=2 | siLastUsedSigOff=3}

Example:

```
val = SR1.Preferences.SignalInit
SR1.Preferences.SignalInit = siSR1Default
```

Description: Generator signal initialization method.



Analog Generator Max. Output

40.000 Vp

AnlgGenMaxVolt

Command Syntax: SR1.Preferences.AnlgGenMaxVolt(*ValueUnit*)

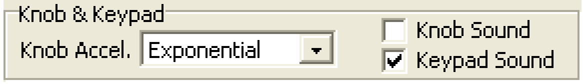
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Preferences.AnlgGenMaxVolt("VP")
SR1.Preferences.AnlgGenMaxVolt("VP") = 40
```

Description: Maximum analog generator output voltage.



Knob & Keypad

Knob Accel. Exponential

□ Knob Sound

☑ Keypad Sound

KeypadSound

Command Syntax: SR1.Preferences.KeypadSound

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.KeypadSound
SR1.Preferences.KeypadSound = False
```

Description: On/off status of the sound made when a keypad key is pressed.

KnobAccel

Command Syntax: SR1.Preferences.KnobAccel

Command Argument(s):

Property: knobAcc enumeration {kaExponential=0 | kaPowerLaw=1 | kaCursorPos=2}

Example:

```
val = SR1.Preferences.KnobAccel
SR1.Preferences.KnobAccel = kaExponential
```

Description: Knob acceleration algorithm.

KnobSound

Command Syntax: SR1.Preferences.KnobSound

Command Argument(s):

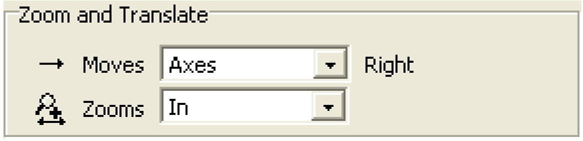
Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.KnobSound
SR1.Preferences.KnobSound = False
```

Description: On/off status of the sound made when the knob is turned.

Display Preference Commands



Zoom and Translate

Moves Axes Right

Zooms In

ShiftMode

Command Syntax: SR1.Preferences.ShiftMode

Command Argument(s):

Property: shiftPref enumeration {spShiftData=0 | spShiftAxes=1}

Example:

```
val = SR1.Preferences.ShiftMode
SR1.Preferences.ShiftMode = spShiftData
```

Description: Sense of the translate scaling control (moves data vs. moves axes).

ZoomMode

Command Syntax: SR1.Preferences.ZoomMode

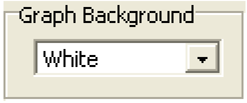
Command Argument(s):

Property: zoomPref enumeration {zpZoomData=0 | zpZoomAxes=1}

Example:

```
val = SR1.Preferences.ZoomMode
SR1.Preferences.ZoomMode = zpZoomData
```

Description: Sense of the zoom scaling controls (zooms data vs. zooms axes).



Graph Background

White

GraphBackgroundColor

Command Syntax: SR1.Preferences.GraphBackgroundColor

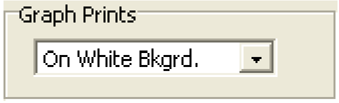
Command Argument(s):

Property: backgroundColor enumeration {bcWhite=0 | bcBlack=1}

Example:

```
val = SR1.Preferences.GraphBackgroundColor
SR1.Preferences.GraphBackgroundColor = bcWhite
```

Description: Default graph background color (black or white).



Graph Prints

On White Bkgrd.

GraphPrint

Command Syntax: SR1.Preferences.GraphPrint

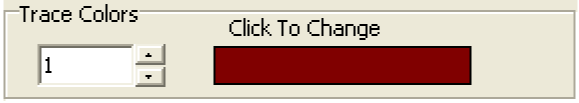
Command Argument(s):

Property: printBackground enumeration {pbAsDisplayed=0 | pbWhiteBackground=1}

Example:

```
val = SR1.Preferences.GraphPrint
SR1.Preferences.GraphPrint = pbAsDisplayed
```

Description: Sets whether graphs print with the current or white background.



Trace Colors

1

GetTraceColor

Command Syntax: SR1.Preferences.GetTraceColor(*Index*, *BGColor*)

Command Argument(s): *Index* as Integer
BGColor as backgroundColor enumeration {bcWhite=0 | bcBlack=1}

Return Value: *Color* as Integer

Example:

```
Color = SR1.Preferences.GetTraceColor(1, bcWhite)
```

Description: Returns the color used for the Indexth trace with the specified graph background color. (The set of colors is different depending on whether a white or black background is used.)

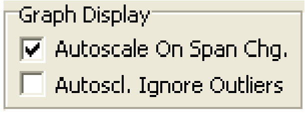
SetTraceColor

Command Syntax: SR1.Preferences.SetTraceColor(*Index*, *Color*)

Command Argument(s): *Index* as Integer
Color as Integer

Example:
```
Call SR1.Preferences.SetTraceColor(1, 4532)
```

Description: Sets the color used for the Indexth trace with the current graph background color. (The set of colors is different depending on whether a white or black background is used.)



Graph Display

☑ Autoscale On Span Chg.

□ Autoscl. Ignore Outliers

AutoSpanFFT

Command Syntax: SR1.Preferences.AutoSpanFFT

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Preferences.AutoSpanFFT
SR1.Preferences.AutoSpanFFT = False
```

Description: Sets whether traces displaying live FFT data will automatically autoscale when the FFT analysis range is changed.

GraphAutosclIgnoreOutliers

Command Syntax: SR1.Preferences.GraphAutosclIgnoreOutliers

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Preferences.GraphAutosclIgnoreOutliers
SR1.Preferences.GraphAutosclIgnoreOutliers = False
```

Description: Sets whether the graph autoscale algorithm will ignore isolated outlier points.

Trace Initialization

Last Used

TraceInit

Command Syntax: SR1.Preferences.TraceInit

Command Argument(s):

Property: TraceInitialization enumeration {tiSR1Default=0 | tiLastUsed=1}

Example:

```
val = SR1.Preferences.TraceInit
SR1.Preferences.TraceInit = tiSR1Default
```

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: SR1.Preferences.ScreenSize

Command Argument(s):

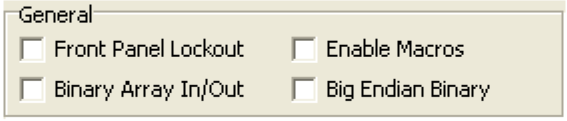
Property: ScreenSizes enumeration {ss100pct=0 | ss122pct=1 | ss136pct=2 | ss152pct=3}

Example:

```
val = SR1.Preferences.ScreenSize
SR1.Preferences.ScreenSize = ss100pct
```

Description: Value of the panel size parameter.

Remote Interface Commands



General

□ Front Panel Lockout □ Enable Macros

□ Binary Array In/Out □ Big Endian Binary

RemotePanelLockout

Command Syntax: SR1.Preferences.RemotePanelLockout

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.RemotePanelLockout
SR1.Preferences.RemotePanelLockout = False
```

Description: Sets whether the instrument implements "Remote" mode, i.e. front panel lockout.

RemoteArrayIO

Command Syntax: SR1.Preferences.RemoteArrayIO

Command Argument(s):

Property: binaryArray enumeration {baASCII=0 | baBinary=1}

Example:

```
val = SR1.Preferences.RemoteArrayIO
SR1.Preferences.RemoteArrayIO = baASCII
```

Description: Queries whether commands which return arrays will return ascii or binary values.

RemoteBinaryEndian

Command Syntax: SR1.Preferences.RemoteBinaryEndian

Command Argument(s):

Property: binaryEndian enumeration {beLittleEndian=0 | beBigEndian=1}

Example:

```
val = SR1.Preferences.RemoteBinaryEndian
SR1.Preferences.RemoteBinaryEndian = beLittleEndian
```

Description: Big/Little Endian format that binary arrays will be sent in.

RemoteEnableMacro

Command Syntax: SR1.Preferences.RemoteEnableMacro

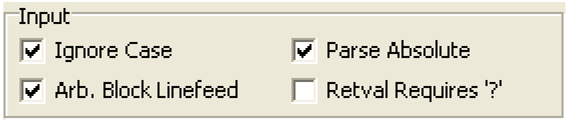
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.RemoteEnableMacro
SR1.Preferences.RemoteEnableMacro = False
```

Description: Enabled status of GPIB macros.



Input

☑ Ignore Case ☑ Parse Absolute

☑ Arb. Block Linefeed □ Retval Requires '?'

RemoteArbBlockLF

Command Syntax: SR1.Preferences.RemoteArbBlockLF

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.RemoteArbBlockLF
SR1.Preferences.RemoteArbBlockLF = False
```

Description: Sets whether a LF (linefeed) character is required to terminate input arbitrary block data.

RemoteIgnoreCase

Command Syntax: SR1.Preferences.RemoteIgnoreCase

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.RemoteIgnoreCase
SR1.Preferences.RemoteIgnoreCase = False
```

Description: Queries whether case is ingored when parsing received gpib commands.

RemoteParseMode

Command Syntax: SR1.Preferences.RemoteParseMode

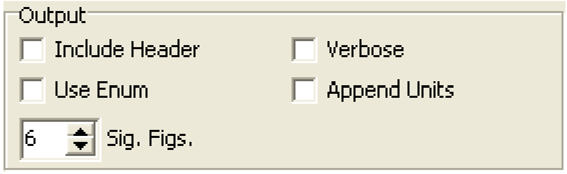
Command Argument(s):

Property: parseMode enumeration {pmParseRelative=0 | pmParseAbsolute=1}

Example:

```
val = SR1.Preferences.RemoteParseMode
SR1.Preferences.RemoteParseMode = pmParseRelative
```

Description: Queries whether GPIB object descriptions must be include the complete path from the root or are relative to the last referenced object.



Output

□ Include Header　□ Verbose

□ Use Enum　□ Append Units

6 Sig. Figs.

RemoteAppendUnits

Command Syntax: SR1.Preferences.RemoteAppendUnits

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.RemoteAppendUnits
SR1.Preferences.RemoteAppendUnits = False
```

Description: Sets whether unit strings will be appended to the responses to GPIB queries.

RemoteSigFig

Command Syntax: SR1.Preferences.RemoteSigFig

Command Argument(s):

Property: Integer

Example:
```
val = SR1.Preferences.RemoteSigFig
SR1.Preferences.RemoteSigFig = Value
```

Description: Number of significant figures used in responses to GPIB queries.

RemoteUseEnum

Command Syntax: SR1.Preferences.RemoteUseEnum

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Preferences.RemoteUseEnum
SR1.Preferences.RemoteUseEnum = False
```

Description: Sets whether responses will be sent as integers or as ascii enumerations.

RemoteUseHeader

Command Syntax: SR1.Preferences.RemoteUseHeader

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Preferences.RemoteUseHeader
SR1.Preferences.RemoteUseHeader = False
```

Description: Sets whether responses will include the command header.

RemoteVerbose

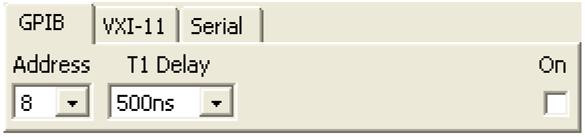
Command Syntax: SR1.Preferences.RemoteVerbose

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Preferences.RemoteVerbose
SR1.Preferences.RemoteVerbose = False
```

Description: On/off status of verbose mode.



Address: 8
T1 Delay: 500ns
On: □

GPIBAddress

Command Syntax: SR1.Preferences.GPIBAddress

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Preferences.GPIBAddress
SR1.Preferences.GPIBAddress = 8
```

Description: Primary GPIB address of the instrument.

GPIBEnabled

Command Syntax: SR1.Preferences.GPIBEnabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.GPIBEnabled
SR1.Preferences.GPIBEnabled = False
```

Description: Sets whether the GPIB (IEEE-488) interface is enabled.

GPIBT1Delay

Command Syntax: SR1.Preferences.GPIBT1Delay

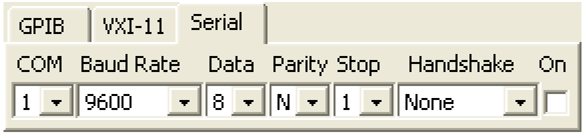
Command Argument(s):

Property: t1Delay enumeration {td1100ns=0 | td500ns=1 | td350ns=2}

Example:

```
val = SR1.Preferences.GPIBT1Delay
SR1.Preferences.GPIBT1Delay = td1100ns
```

Description: GPIB T1 delay value.



COM	Baud Rate	Data	Parity	Stop	Handshake	On
1	9600	8	N	1	None	□

SerialBitRate

Command Syntax: SR1.Preferences.SerialBitRate

Command Argument(s):

Property: bitRate enumeration {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:

```
val = SR1.Preferences.SerialBitRate
SR1.Preferences.SerialBitRate = br57600
```

Description: Speed (bit rate) of the rear panel serial interface.

SerialDataBits

Command Syntax: SR1.Preferences.SerialDataBits

Command Argument(s):

Property: dataBits enumeration {db7=0 | db8=1}

Example:

```
val = SR1.Preferences.SerialDataBits
SR1.Preferences.SerialDataBits = db8
```

Description: Number of data bits for the serial interface.

SerialEnabled

Command Syntax: SR1.Preferences.SerialEnabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.SerialEnabled
SR1.Preferences.SerialEnabled = False
```

Description: Enabled status of the serial interface.

SerialHandshake

Command Syntax: SR1.Preferences.SerialHandshake

Command Argument(s):

Property: handshake enumeration {hsNone=0 | hsRTSCTS=1 | hsDTRDSR=2 | hsXonXoff=3}

Example:

```
val = SR1.Preferences.SerialHandshake
SR1.Preferences.SerialHandshake = hsNone
```

Description: Serial interface handshake mode.

SerialParity

Command Syntax: SR1.Preferences.SerialParity

Command Argument(s):

Property: parity enumeration {ptNone=0 | ptOdd=1 | ptEven=2}

Example:

```
val = SR1.Preferences.SerialParity
SR1.Preferences.SerialParity = ptNone
```

Description: serial interface parity selection.

SerialPort

Command Syntax: SR1.Preferences.SerialPort

Command Argument(s):

Property: portEnum enumeration {peCOM1=0 | peCOM2=1 | peCOM3=2 | peCOM4=3}

Example:

```
val = SR1.Preferences.SerialPort
SR1.Preferences.SerialPort = peCOM1
```

Description: 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: SR1.Preferences.SerialStopBits

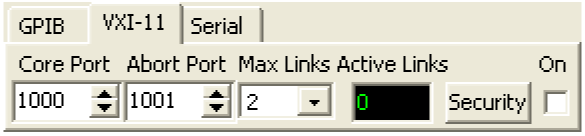
Command Argument(s):

Property: stopBits enumeration {sb1=0 | sb2=1}

Example:

```
val = SR1.Preferences.SerialStopBits
SR1.Preferences.SerialStopBits = sb1
```

Description: number of stop bits for the serial interface.



Core Port 1000
Abort Port 1001
Max Links 2
Active Links 0
On □

VXIAbortPort

Command Syntax: SR1.Preferences.VXIAbortPort

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Preferences.VXIAbortPort
SR1.Preferences.VXIAbortPort = 1001
```

Description: Abort Port number for the VXI11 interface.

VXIActiveLinksRdg

Command Syntax: SR1.Preferences.VXIActiveLinksRdg

Command Argument(s):

Property: Integer (read-only)

Example:

```
val = SR1.Preferences.VXIActiveLinksRdg
```

Description: Number of sessions currently active on the VXI-11 interface.

VXIAllowList

Command Syntax: SR1.Preferences.VXIAllowList

Command Argument(s):

Property: String

Example:

```
val = SR1.Preferences.VXIAllowList
SR1.Preferences.VXIAllowList = "123.45.*.*,192.168.0.1"
```

Description: Sets the list of "allowed" IP addresses that can connect via VXI-11. Each line in the list is separated by a comma.

VXICheckDeniedFirst

Command Syntax: SR1.Preferences.VXICheckDeniedFirst

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.VXICheckDeniedFirst
SR1.Preferences.VXICheckDeniedFirst = False
```

Description: Queries whether the denied list is checked before the allowed list when determining whether an IP address is allowed to connect.

VXICorePort

Command Syntax: SR1.Preferences.VXICorePort

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Preferences.VXICorePort
SR1.Preferences.VXICorePort = 1000
```

Description: Value of the core VXI-11 port.

VXIDenyList

Command Syntax: SR1.Preferences.VXIDenyList

Command Argument(s):

Property: String

Example:

```
val = SR1.Preferences.VXIDenyList
SR1.Preferences.VXIDenyList = "75.101.142.50,88.65.*.*"
```

Description: List of denied IP addresses. Each entry is separated by a comma.

VXIEnabled

Command Syntax: SR1.Preferences.VXIEnabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.VXIEnabled
SR1.Preferences.VXIEnabled = False
```

Description: Enabled status of the VXI-11 interface.

VXIMaxLinks

Command Syntax: SR1.Preferences.VXIMaxLinks

Command Argument(s):

Property: maxLinks enumeration {ml1=0 | ml2=1 | ml5=2 | ml10=3 | mlMax=4}

Example:

```
val = SR1.Preferences.VXIMaxLinks
SR1.Preferences.VXIMaxLinks = ml1
```

Description: Maximum number of simultaneous connections allowed over the VXI-11 interface.

VXIPassword

Command Syntax: SR1.Preferences.VXIPassword

Command Argument(s):

Property: String

Example:

```
val = SR1.Preferences.VXIPassword
SR1.Preferences.VXIPassword = Value
```

Description: VXI-11 connection password.

VXIRequirePassword

Command Syntax: SR1.Preferences.VXIRequirePassword

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Preferences.VXIRequirePassword
SR1.Preferences.VXIRequirePassword = False
```

Description: Sets whether a password is required to connect to the VXI-11 interface.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.14 Scripting

Object:	**SR1.Scripting**
Object Argument(s):	None
Description:	Commands related to the local scripting environment of SR1.

FileName

Command Syntax: SR1.Scripting.FileName

Command Argument(s):

Property: String (read-only)

Example:

```
val = SR1.Scripting.FileName
```

Description: Filename of the current script file.

IsModified

Command Syntax: SR1.Scripting.IsModified

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Scripting.IsModified
```

Description: Queries whether the script has been modified since the last save.

Language

Command Syntax: SR1.Scripting.Language

Command Argument(s):

Property: ScriptLang enumeration {scVBScript=0 | scJScript=1}

Example:

```
val = SR1.Scripting.Language
SR1.Scripting.Language = scVBScript
```

Description: Script language selection.

ShowUI

Command Syntax: SR1.Scripting.ShowUI

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Scripting.ShowUI
SR1.Scripting.ShowUI = False
```

Description: Sets whether the right-hand panels of the scripting window display the SR1 Basic user interface.

Terminate

Command Syntax: SR1.Scripting.Terminate

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Scripting.Terminate
SR1.Scripting.Terminate = False
```

Description: Value of the script terminate flag. The script terminate flag can be checked by scripts as a signal to terminate processing.

ClearLines

Command Syntax: SR1.Scripting.ClearLines()

Command Argument(s): None

Example:

```
Call SR1.Scripting.ClearLines()
```

Description: Clears all lines of the current script.

IncludeScript

Command Syntax: SR1.Scripting.IncludeScript(*FileName*, *Location*)

Command Argument(s): *FileName* as String
Location as IncludeLocation enumeration {ilUserScript=0 | ilEventScript=1}

Example:

```
Call SR1.Scripting.IncludeScript("MyScript.vbs", ilUserS
```

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: SR1.Scripting.Load(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.Scripting.Load("MyScript.vbs")
```

Description: Loads the script with the specified filename.

New

Command Syntax: SR1.Scripting.New()

Command Argument(s): None

Example:

```
Call SR1.Scripting.New()
```

Description: Clears the current script and creates a new, empty script with the filename "newscript.vbs".

Run

Command Syntax: SR1.Scripting.Run(*Timeout*)

Command Argument(s): *Timeout* as Integer

Example: `Call SR1.Scripting.Run(Value)`

Description: Runs the current script.

Save

Command Syntax: SR1.Scripting.Save()

Command Argument(s): None

Example: `Call SR1.Scripting.Save()`

Description: Saves the current script to the current scripting file.

SaveAs

Command Syntax: SR1.Scripting.SaveAs(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Scripting.SaveAs("Value")`

Description: Saves the current script to a new scripting file and make that file the current file.

StopEvents

Command Syntax: SR1.Scripting.StopEvents()

Command Argument(s): None

Example: `Call SR1.Scripting.StopEvents()`

Description: Stops script processing of events.

Scripting Window Form Commands

OpenForm

Command Syntax: SR1.Scripting.OpenForm()

Command Argument(s): None

Example: `Call SR1.Scripting.OpenForm()`

Description: Opens a scripting window on the current page of the page control. Only one scripting form is allowed at any time.

CloseForm

Command Syntax: SR1.Scripting.CloseForm()

Command Argument(s): None

Example: `Call SR1.Scripting.CloseForm()`

Description: Closes the scripting window. This does not stop a running script, nor does it clear the contents of the current script.

FormID

Command Syntax: SR1.Scripting.FormID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Scripting.FormID()`

Description: Returns the FormID of the scripting window.

Scripting Log Commands

OpenScriptLogForm

Command Syntax: SR1.Scripting.OpenScriptLogForm()

Command Argument(s): None

Example: `Call SR1.Scripting.OpenScriptLogForm()`

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

OpenScriptLogFormwID

Command Syntax: SR1.Scripting.OpenScriptLogFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Scripting.OpenScriptLogFormwID()`

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: SR1.Scripting.ScriptLogFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Scripting.ScriptLogFormCount()`

Description: Counts the number of script log forms on all pages of the page control.

ScriptLogFormID

Command Syntax: SR1.Scripting.ScriptLogFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.Scripting.ScriptLogFormID(1)`

Description: Returns the FormID of the Indexth script log form.

CloseScriptLogForm

Command Syntax: SR1.Scripting.CloseScriptLogForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.Scripting.CloseScriptLogForm(5)`

Description: Closes the scipt log form with the given FormID.

CloseScriptLogForms

Command Syntax: SR1.Scripting.CloseScriptLogForms()

Command Argument(s): None

Example: `Call SR1.Scripting.CloseScriptLogForms()`

Description: Closes all script log forms on all pages of the page control.

WriteLine

Command Syntax: SR1.Scripting.WriteLine(*Text*)

Command Argument(s): *Text* as String

Example: `Call SR1.Scripting.WriteLine("Output=100")`

Description: Writes a new line to the script log form.

PrintLog

Command Syntax: SR1.Scripting.PrintLog()

Command Argument(s): None

Example: `Call SR1.Scripting.PrintLog()`

Description: Prints the contents of the script log form to the currently configured SR1 printer.

2.4.15 Displays

Object:	**SR1.Displays**
Object Argument(s):	None
Description:	Commands related to SR1's collection of displays

Load

Command Syntax: SR1.Displays.Load(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Displays.Load("DisplayFile.XML")`

Description: Loads the indicated display file.

SaveAll

Command Syntax: SR1.Displays.SaveAll(*FileName*, *GraphData*)

Command Argument(s): *FileName* as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Displays.SaveAll("MyFile.XML", gdNeverSave)`

Description: Saves all displays on all pages of the page control to the indictaed 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: SR1.Displays.DeleteAll()

Command Argument(s): None

Example: `Call SR1.Displays.DeleteAll()`

Description: Deletes all displays on all pages of the page control.

Bar Chart Commands

NewBar

Command Syntax: SR1.Displays.NewBar()

Command Argument(s): None

Return Value: *BarID* as Integer

Example: `BarID = SR1.Displays.NewBar()`

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 SR1.Displays.Bar(BarID).

BarCount

Command Syntax: SR1.Displays.BarCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Displays.BarCount()`

Description: Returns the total number of bar charts found on all pages of the page control.

BarItem

Command Syntax: SR1.Displays.BarItem(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Bar* as Object

Example: `Bar = SR1.Displays.BarItem(0)`

Description: Returns the objectID of the Indexth bar chart. The properties of the chart can subsequently be manipulated using the object SR1.Displays.Bar(BarID).

Digitizer Display Commands

NewDigitizerDisplay

Command Syntax: SR1.Displays.NewDigitizerDisplay()

Command Argument(s): None

Return Value: *DigitizerID* as Integer

Example: `DigitizerID = SR1.Displays.NewDigitizerDisplay()`

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 SR1.Displays.DigitzerDisplay (digitizerID).

DigitizerDisplayCount

Command Syntax: SR1.Displays.DigitizerDisplayCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Displays.DigitizerDisplayCount()`

Description: Returns the total number of digitizer displays on all pages of the page control.

DigitizerDisplayItem

Command Syntax: SR1.Displays.DigitizerDisplayItem(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *DigitizerDisplay* as Object

Example: `DigitizerDisplay = SR1.Displays.DigitizerDisplayItem(0)`

Description: Returns the objectID of the Indexth digitizer display. The properties of the display can subsequently be manipulated using the object SR1.Displays.DigitzerDisplay (digitizerID).

Graph Commands

NewGraph

Command Syntax: SR1.Displays.NewGraph()

Command Argument(s): None

Return Value: *GraphID* as Integer

Example: `GraphID = SR1.Displays.NewGraph()`

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 SR1.Displays.Graph(GraphID).

GraphCount

Command Syntax: SR1.Displays.GraphCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.Displays.GraphCount()`

Description: Returns the total number of graphs on all pages of the page control.

GraphItem

Command Syntax: SR1.Displays.GraphItem(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Graph* as Object

Example: `Graph = SR1.Displays.GraphItem(0)`

Description: Returns the objectID of the Indexth graph. The properties of the graph can subsequently be manipulated using the object SR1.Displays.Graph(GraphID).

2.4.15.1 Graph

Object:	**SR1.Displays.Graph(*i*)**
Object Argument(s):	*i* as Integer
Description:	Commands related to the graph with objectID i.

Close

Command Syntax: SR1.Displays.Graph(i).Close()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).Close()`

Description: Closes the graph.

BackgroundColor

Command Syntax: SR1.Displays.Graph(i).BackgroundColor

Command Argument(s):

Property: backgroundColor enumeration {bcWhite=0 | bcBlack=1}

Example:
```
val = SR1.Displays.Graph(1).BackgroundColor
SR1.Displays.Graph(1).BackgroundColor = bcWhite
```

Description: Sets whether the graph has a white or black background.

Title

Command Syntax: SR1.Displays.Graph(i).Title

Command Argument(s):

Property: String

Example:
```
val = SR1.Displays.Graph(1).Title
SR1.Displays.Graph(1).Title = "Filter Response"
```

Description: Graph title.

FormID

Command Syntax: SR1.Displays.Graph(i).FormID()

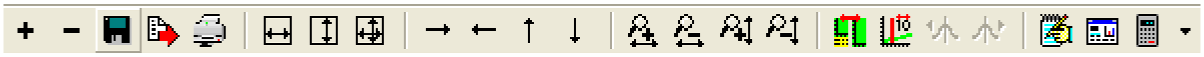
Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Displays.Graph(1).FormID()`

Description: FormID for the graph. The FormID can the 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: SR1.Displays.Graph(i).AddEQTrace(*FileName*, *Type*)

Command Argument(s): *FileName* as String
Type as EQTraceType enumeration {eqAmp=0 | eqAmpInvt=1 | eqPhase=2}

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).AddEQTrace("Aweight.e`

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: SR1.Displays.Graph(i).AddTrace(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).AddTrace(msA0FFTspect`

Description: Adds a trace to the graph corresponding to the specified measurement and returns the ID of the newly created trace.

DeleteAllTraces

Command Syntax: SR1.Displays.Graph(i).DeleteAllTraces()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).DeleteAllTraces()`

Description: Deletes all traces on the graph.

DeleteSweepTraces

Command Syntax: SR1.Displays.Graph(i).DeleteSweepTraces(*TraceID*)

Command Argument(s): *TraceID* as Integer

Example: `Call SR1.Displays.Graph(1).DeleteSweepTraces(102)`

Description: Deletes all sweep traces with the same sweep data as Trace(TraceID).

DeleteTrace

Command Syntax: SR1.Displays.Graph(i).DeleteTrace(*TraceID*)

Command Argument(s): *TraceID* as Integer

Example: `Call SR1.Displays.Graph(1).DeleteTrace(104)`

Description: Deletes the trace with the specific ID.

DeleteUnusedTraces

Command Syntax: SR1.Displays.Graph(i).DeleteUnusedTraces()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).DeleteUnusedTraces()`

Description: Deletes traces that are unchecked (not displayed).

Save

Command Syntax: SR1.Displays.Graph(i).Save(*FileName*, *GraphData*)

Command Argument(s): *FileName* as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Displays.Graph(1).Save("MyGraphFile.XML", gdNev`

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: SR1.Displays.Graph(i).ExportData(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Displays.Graph(1).ExportData("MyData.txt")`

Description: Exports graph data to a text file with the given file name. Trace data for all traces are saved.

ExportGraph

Command Syntax: SR1.Displays.Graph(i).ExportGraph(*Kind*, *FileName*)

Command Argument(s): *Kind* as FileKind enumeration {ftBitMap=0 | ftEnMetaFile=1 | ftJpeg=2}
FileName as String

Example: `Call SR1.Displays.Graph(1).ExportGraph(0, "GraphFile.BMP`

Description: Exports the graph to the specified graphics file type and filename.

Print

Command Syntax: SR1.Displays.Graph(i).Print()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).Print()`

Description: Prints the graph to the currently configured default printer.

AutoScale

Command Syntax: SR1.Displays.Graph(i).AutoScale()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).AutoScale()`

Description: Autoscales the X and Y axes of the active trace (or all traces if ApplyAll is true) to fit the data.

AutoScaleX

Command Syntax: SR1.Displays.Graph(i).AutoScaleX()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).AutoScaleX()`

Description: Autoscales the X-axis of the active trace (or all traces if ApplyAll is true) to fit the data.

AutoScaleY

Command Syntax: SR1.Displays.Graph(i).AutoScaleY()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).AutoScaleY()`

Description: Autoscales the Y-axis of the active trace (or all traces if ApplyAll is true) to fit the data.

ShiftX

Command Syntax: SR1.Displays.Graph(i).ShiftX(*Direction*)

Command Argument(s): *Direction* as Boolean {xLeft=0 | xRight=1}

Example: `Call SR1.Displays.Graph(1).ShiftX(xLeft)`

Description: Shifts the data (or axes) of the active trace (or all traces if ApplyAll is true) 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: SR1.Displays.Graph(i).ShiftY(*Direction*)

Command Argument(s): *Direction* as Boolean {yDown=0 | yUp=1}

Example: `Call SR1.Displays.Graph(1).ShiftY(yDown)`

Description: Shifts the data (or axes) of the active trace (or all traces if ApplyAll is true) 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: SR1.Displays.Graph(i).ZoomX(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.Graph(1).ZoomX(zmOut)`

Description: Zooms the X-axis of the active trace (or all traces if ApplyAll is true) 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: SR1.Displays.Graph(i).ZoomY(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.Graph(1).ZoomY(zmOut)`

Description: Zooms the Y-axis of the active trace (or all traces if ApplyAll is true) 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: SR1.Displays.Graph(i).Maximize

Command Argument(s):

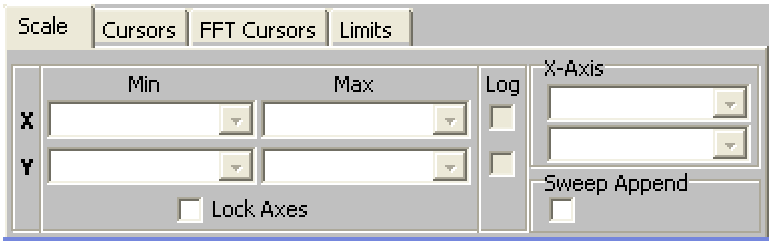
Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(1).Maximize
SR1.Displays.Graph(1).Maximize = False
```

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" (ApplyAll) parameter. To set or query the properties of a specific trace regardless of the state of "Lock Axes" use the SR1.Displays.Graph(i).Trace(j) object.

xAxis

Command Syntax: SR1.Displays.Graph(i).xAxis(*Value*)

Command Argument(s): *Value* as XAxisType enumeration {xaIndex=0 | xaTime=1 | xaSweep=2}

Example:

```
Call SR1.Displays.Graph(1).xAxis(xaIndex)
```

Description: Sets the X-axis selection for scalar traces.

Xlog

Command Syntax: SR1.Displays.Graph(i).Xlog(*On*)

Command Argument(s): *On* as Boolean {False=0 | True=1}

Example:

```
Call SR1.Displays.Graph(1).Xlog(False)
```

Description: Sets logarithmic X-axis on or off.

Xmax

Command Syntax: SR1.Displays.Graph(i).Xmax(*ValueUnit*, *Value*)

Command Argument(s): *ValueUnit* as String
Value as Double

Example:

```
Call SR1.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: SR1.Displays.Graph(i).Xmin(*ValueUnit*, *Value*)

Command Argument(s): *ValueUnit* as String
Value as Double

Example: `Call SR1.Displays.Graph(1).Xmin("Hz", Value)`

Description: Sets the minimum X-value. Send the empty string "" to set the value in current units.

xScaleMode

Command Syntax: SR1.Displays.Graph(i).xScaleMode(*Value*)

Command Argument(s): *Value* as XAxisScalingType enumeration {xaFixed=0 | xaPan=1 | xaFull=2}

Example: `Call SR1.Displays.Graph(1).xScaleMode(xaFixed)`

Description: Sets the scaling mode for scalar traces.

Ylog

Command Syntax: SR1.Displays.Graph(i).Ylog(*On*)

Command Argument(s): *On* as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.Graph(1).Ylog(False)`

Description: Sets logarithmic Y-axis on or off.

Ymax

Command Syntax: SR1.Displays.Graph(i).Ymax(*ValueUnit*, *Value*)

Command Argument(s): *ValueUnit* as String
Value as Double

Example: `Call SR1.Displays.Graph(1).Ymax("Vp", 100.0)`

Description: Sets the maximum Y value.

Ymin

Command Syntax: SR1.Displays.Graph(i).Ymin(*ValueUnit*, *Value*)

Command Argument(s): *ValueUnit* as String
Value as Double

Example: `Call SR1.Displays.Graph(1).Ymin("Vrms", 0.0)`

Description: Sets the minimum Y value.

Append

Command Syntax: SR1.Displays.Graph(i).Append

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(1).Append
SR1.Displays.Graph(1).Append = False
```

Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

ApplyAll

Command Syntax: SR1.Displays.Graph(i).ApplyAll

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(1).ApplyAll
SR1.Displays.Graph(1).ApplyAll = False
```

Description: Sets whether zoom, shift, or axes changes affect the currently selected trace, or all compatible traces.

Trace Commands

☑ A1:FFT1:Power Spectrum A
☑ A1:FFT1:Time Record A
☑ *Offline Trace*

CopyOfflineSweeps

Command Syntax: SR1.Displays.Graph(i).CopyOfflineSweeps()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).CopyOfflineSweeps()`

Description: Copies any live sweep traces offline.

CopyOfflineTrace

Command Syntax: SR1.Displays.Graph(i).CopyOfflineTrace(*TraceID*)

Command Argument(s): *TraceID* as Integer

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).CopyOfflineTrace(101)`

Description: Copies the specified trace to a new offline trace and returns the traceID of the newly created trace.

CopyTrace

Command Syntax: SR1.Displays.Graph(i).CopyTrace(*TraceID*)

Command Argument(s): *TraceID* as Integer

Example: `Call SR1.Displays.Graph(1).CopyTrace(101)`

Description: Copies the specified trace to the clipboard.

CutTrace

Command Syntax: SR1.Displays.Graph(i).CutTrace(*TraceID*)

Command Argument(s): *TraceID* as Integer

Example: `Call SR1.Displays.Graph(1).CutTrace(101)`

Description: Copy the specified trace to the clipboard and then deletes the trace from the graph.

GoOffline

Command Syntax: SR1.Displays.Graph(i).GoOffline()

Command Argument(s): None

Example: `Call SR1.Displays.Graph(1).GoOffline()`

Description: Makes all traces of the graph offline.

IsTraceChecked

Command Syntax: SR1.Displays.Graph(i).IsTraceChecked(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Checked* as Boolean {False=0 | True=1}

Example: `Checked = SR1.Displays.Graph(1).IsTraceChecked(0)`

Description: Queries whether the Indexth trace is checked or not.

TraceCheck

Command Syntax: SR1.Displays.Graph(i).TraceCheck(*Index*, *Check*)

Command Argument(s): *Index* as Integer
Check as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.Graph(1).TraceCheck(3, False)`

Description: Checks (or un-checks) the Indexth trace.

LoadTrace

Command Syntax: SR1.Displays.Graph(i).LoadTrace(*FileName*)

Command Argument(s): *FileName* as String

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).LoadTrace("MyTraceFil`

Description: Loads the trace stored in the specified file and returns the traceID of the newly created trace.

PasteTrace

Command Syntax: SR1.Displays.Graph(i).PasteTrace()

Command Argument(s): None

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).PasteTrace()`

Description: Pastes the trace currently stored in the clipboard into the graph and returns the traceID of the new trace.

SaveTrace

Command Syntax: SR1.Displays.Graph(i).SaveTrace(*TraceID*, *FileName*, *GraphData*)

Command Argument(s): *TraceID* as Integer
FileName as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Displays.Graph(1).SaveTrace(101, "MyFile.XML",`

Description: Saves the the trace with the griven ID into the specified file. The GraphData argument determines whether data will be saved with the trace.

SelectedTrace

Command Syntax: SR1.Displays.Graph(i).SelectedTrace()

Command Argument(s): None

Return Value: *Trace* as Object

Example: `Trace = SR1.Displays.Graph(1).SelectedTrace()`

Description: Returns the currently selected trace object. Parameters of the Trace may be directly accessed from the object,
e.g. Trace.yMax("dB") = -20

SelectTrace

Command Syntax: SR1.Displays.Graph(i).SelectTrace(*Trace*)

Command Argument(s): *Trace* as Object

Example: `Call SR1.Displays.Graph(1).SelectTrace(Trace)`

Description: Causes the trace object to be selected.

TraceCount

Command Syntax: SR1.Displays.Graph(i).TraceCount()

Command Argument(s): None

Return Value: *NumTraces* as Integer

Example: `NumTraces = SR1.Displays.Graph(1).TraceCount()`

Description: Returns the total number of traces in the graph.

TraceItem

Command Syntax: SR1.Displays.Graph(i).TraceItem(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Trace* as Object

Example: `Trace = SR1.Displays.Graph(1).TraceItem(0)`

Description: Returns a string containing the traceID of the Indexth trace of the graph.Properties of the the selected trace in the example above, for instance, could then be accessed using the SR1.Displays:Graph(i):Trace(101).

2.4.15.1.1 Graph Trace

Object:	**SR1.Displays.Graph(*i*).Trace(*j*)**
Object Argument(s):	*i* as Integer, *j* as Integer
Description:	Commands related to a single trace of a graph.

Trace Measurement Commands

Offline

Command Syntax: SR1.Displays.Graph(i).Trace(j).Offline

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(1).Trace(101).Offline
SR1.Displays.Graph(1).Trace(101).Offline = False
```

Description: Returns the Online/Offline status of the trace.

GetMeasID

Command Syntax: SR1.Displays.Graph(i).Trace(j).GetMeasID()

Command Argument(s): None

Return Value: *MeasID* as MeasID enumeration {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 |
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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.Displays.Graph(1).Trace(101).GetMeasID()`

Description: Returns the measurement ID associated with the trace.

GetN

Command Syntax: SR1.Displays.Graph(i).Trace(j).GetN()

Command Argument(s): None

Return Value: *N* as Integer

Example:

```
N = SR1.Displays.Graph(1).Trace(101).GetN()
```

Description: Returns the number of data-points in the trace.

GetNumStoredData

Command Syntax: SR1.Displays.Graph(i).Trace(j).GetNumStoredData()

Command Argument(s): None

Return Value: *N* as Integer

Example:

```
N = SR1.Displays.Graph(1).Trace(101).GetNumStoredData()
```

Description: Returns the current storage depth of the trace. This is the number of measurements that have been take since free-run mode was started. Measurements are stored up to the number set by the "SetStorageDepth" command.

SetStorageDepth

Command Syntax: SR1.Displays.Graph(i).Trace(j).SetStorageDepth(*Depth*)

Command Argument(s): *Depth* as Integer

Example:

```
Call SR1.Displays.Graph(1).Trace(101).SetStorageDepth(10
```

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: SR1.Displays.Graph(i).Trace(j).GetStorageDepth()

Command Argument(s): None

Return Value: *Depth* as Integer

Example:

```
Depth = SR1.Displays.Graph(1).Trace(101).GetStorageDepth
```

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: SR1.Displays.Graph(i).Trace(j).RecallStoredData(*Index*)

Command Argument(s): *Index* as Integer

Example:

```
Call SR1.Displays.Graph(1).Trace(101).RecallStoredData(0
```

Description: For vector (FFT-type) traces, recalls the data from the Indexth stored measurement into the trace. An index of 0 corresponds to the oldest stored data.

RecallStoredDataIntoNewTrace

Command Syntax: SR1.Displays.Graph(i).Trace(j).RecallStoredDataIntoNewTrace(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(1).Trace(101).RecallStor`

Description: For vector (FFT-type) traces, recalls the data from the Indexth stored measurement into a new trace and returns the TraceID of the newly created trace..

GetXArray

Command Syntax: SR1.Displays.Graph(i).Trace(j).GetXArray(*XUnit*)

Command Argument(s): *XUnit* as String

Return Value: *XArray* as Array

Example: `XArray = SR1.Displays.Graph(1).Trace(101).GetXArray("")`

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: SR1.Displays.Graph(i).Trace(j).GetXY(*Index*, *XUnit*, *YUnit*)

Command Argument(s): *Index* as Integer
XUnit as String
YUnit as String

Return Value: *XYRdg* as Array

Example: `XYRdg = SR1.Displays.Graph(1).Trace(101).GetXY(0, "", ""`

Description: Returns the X-Y values corresponding to the Indexth point in the trace in the specified units. The empty string ("") specifies the current X or Y units.

GetYArray

Command Syntax: SR1.Displays.Graph(i).Trace(j).GetYArray(*YUnit*)

Command Argument(s): *YUnit* as String

Return Value: *YArray* as Array

Example: `YArray = SR1.Displays.Graph(1).Trace(101).GetYArray("")`

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: SR1.Displays.Graph(i).Trace(j).Rename(*Name*)

Command Argument(s): *Name* as String

Example:
```
Call SR1.Displays.Graph(1).Trace(101).Rename("MyTrace")
```

Description: Changes the name of the trace in the graph trace listing. Only applies to offline traces.

SetColor

Command Syntax: SR1.Displays.Graph(i).Trace(j).SetColor(*Color*)

Command Argument(s): *Color* as Integer

Example:
```
Call SR1.Displays.Graph(1).Trace(101).SetColor(144)
```

Description: Sets the color of the trace.

SetWidth

Command Syntax: SR1.Displays.Graph(i).Trace(j).SetWidth(*Width*)

Command Argument(s): *Width* as Integer

Example:
```
Call SR1.Displays.Graph(1).Trace(101).SetWidth(2)
```

Description: Sets the width of the trace in pixels.

xAxis

Command Syntax: SR1.Displays.Graph(i).Trace(j).xAxis

Command Argument(s):

Property: XAxisType enumeration {xaIndex=0 | xaTime=1 | xaSweep=2}

Example:
```
val = SR1.Displays.Graph(1).Trace(101).xAxis
SR1.Displays.Graph(1).Trace(101).xAxis = xaIndex
```

Description: For stripchart traces, queries the X-axis type.

Xlog

Command Syntax: SR1.Displays.Graph(i).Trace(j).Xlog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Xlog
SR1.Displays.Graph(1).Trace(101).Xlog = False
```

Description: Log/linear status of the trace X-axis.

Xmax

Command Syntax: SR1.Displays.Graph(i).Trace(j).Xmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Xmax("")
SR1.Displays.Graph(1).Trace(101).Xmax("") = 40
```

Description: Maximum displayed X-axis value of trace.

Xmin

Command Syntax: SR1.Displays.Graph(i).Trace(j).Xmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Xmin("Sec")
SR1.Displays.Graph(1).Trace(101).Xmin("Sec") = 0.00000
```

Description: Minimum displayed X-axis value of trace.

xScaleMode

Command Syntax: SR1.Displays.Graph(i).Trace(j).xScaleMode

Command Argument(s):

Property: XAxisScalingType enumeration {xaFixed=0 | xaPan=1 | xaFull=2}

Example:
```
val = SR1.Displays.Graph(1).Trace(101).xScaleMode
SR1.Displays.Graph(1).Trace(101).xScaleMode = xaFixed
```

Description: For stripchart type displays, queries the scaling mode used when new points are added to the trace.

Ylog

Command Syntax: SR1.Displays.Graph(i).Trace(j).Ylog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Ylog
SR1.Displays.Graph(1).Trace(101).Ylog = False
```

Description: Log/linear status of the trace Y-axis.

Ymax

Command Syntax: SR1.Displays.Graph(i).Trace(j).Ymax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Ymax("Vp")
SR1.Displays.Graph(1).Trace(101).Ymax("Vp") = 8.95705E-0
```

Description: Maximum displayed Y-axis value of trace.

Ymin

Command Syntax: SR1.Displays.Graph(i).Trace(j).Ymin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(1).Trace(101).Ymin("Vp")
SR1.Displays.Graph(1).Trace(101).Ymin("Vp") = -2.72367E-
```

Description: Minimum displayed Y-axis value of trace.

AutoScale

Command Syntax: SR1.Displays.Graph(i).Trace(j).AutoScale()

Command Argument(s): None

Example:
```
Call SR1.Displays.Graph(1).Trace(101).AutoScale()
```

Description: Autoscales the trace's X and Y axes.

AutoScaleX

Command Syntax: SR1.Displays.Graph(i).Trace(j).AutoScaleX()

Command Argument(s): None

Example:
```
Call SR1.Displays.Graph(1).Trace(101).AutoScaleX()
```

Description: Autoscales the trace X-axis.

AutoScaleY

Command Syntax: SR1.Displays.Graph(i).Trace(j).AutoScaleY()

Command Argument(s): None

Example:
```
Call SR1.Displays.Graph(1).Trace(101).AutoScaleY()
```

Description: Autoscales the trace Y-axis.

Trace Calculator Commands

Calc2Sigma

Command Syntax: SR1.Displays.Graph(i).Trace(j).Calc2Sigma(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).Calc2Sigm`

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: SR1.Displays.Graph(i).Trace(j).CalcAverage(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*, *Mode*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double
Mode as AvgMode enumeration {amLinear=0 | amRMS=1 | amVariance=2}

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcAvera`

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: SR1.Displays.Graph(i).Trace(j).CalcDifference(*Trace2*)

Command Argument(s): *Trace2* as Object

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcDiffe`

Description: Creates a trace containing the difference between the original trace and the trace referenced by the *Trace2* object. The query returns the ID of the newly created trace.

CalcEQ

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcEQ(*EQFilename*, *Invert*)

Command Argument(s): *EQFilename* as String
Invert as Boolean {False=0 | True=1}

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcEQ("M`

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. The x-axis of the trace must be in frequency.

CalcGroupDelay

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcGroupDelay()

Command Argument(s): None

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcGroup`

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: SR1.Displays.Graph(i).Trace(j).CalcInvert(*XPosUnit*, *XPos*)

Command Argument(s): *XPosUnit* as String
XPos as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcInver`

Description: Creates an inverted trace around the specified X-position and returns the traceID of the newly created trace.

CalcLinearity

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcLinearity(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcLinea`

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: SR1.Displays.Graph(i).Trace(j).CalcMakeArb(*ArbFilename*)

Command Argument(s): *ArbFilename* as String

Return Value: *CalcTraceID* as Integer

Example:
```
CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcMakeA
```

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: SR1.Displays.Graph(i).Trace(j).CalcMakeEQ(*XPosUnit*, *XPos*, *EQFilename*)

Command Argument(s): *XPosUnit* as String
XPos as Double
EQFilename as String

Return Value: *CalcTraceID* as Integer

Example:
```
CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcMakeE
```

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: SR1.Displays.Graph(i).Trace(j).CalcMaximum(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double

Return Value: *CalcTraceID* as Integer

Example:
```
CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcMaxim
```

Description: Returns the traceID of a newly created trace containing the maximum value of the original trace.

CalcMinimum

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcMinimum(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double

Return Value: *CalcTraceID* as Integer

Example:
```
CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcMinim
```

Description: Returns the traceID of a newly created trace containing the minimum value of the original trace.

CalcMultiply

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcMultiply(*Trace2*)

Command Argument(s): *Trace2* as Object

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcMulti`

Description: Returns the traceID of a new trace containing the product of the original trace with the trace specified by the *Trace2* object.

CalcNormalize

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcNormalize(*XPosUnit*, *XPos*, *YValUnit*, *YVal*)

Command Argument(s): *XPosUnit* as String
XPos as Double
YValUnit as String
YVal as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcNorma`

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: SR1.Displays.Graph(i).Trace(j).CalcParametric(*Trace2*)

Command Argument(s): *Trace2* as Object

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcParam`

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* object as the *X* data. Trace2 must be at least as long of the original trace.

CalcRatio

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcRatio(*Trace2*)

Command Argument(s): *Trace2* as Object

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcRatio`

Description: Returns the traceID of a new trace whose data is the ratio of the original trace to the trace specified by the *Trace2* object.

CalcSmooth

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcSmooth(*Mode*, *SmoothParam*, *Repeat*)

Command Argument(s): *Mode* as SmoothAlgorithm enumeration {saBoxAvg=0 | saBinomial=1 | saSavitzkyGolay=2}
SmoothParam as Integer
Repeat as Integer

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcSmoot`

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: SR1.Displays.Graph(i).Trace(j).CalcTrim(*FromXUnit*, *FromX*, *ToXUnit*, *ToX*)

Command Argument(s): *FromXUnit* as String
FromX as Double
ToXUnit as String
ToX as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcTrim(`

Description: Returns the traceID of a new trace containing the original trace data but only including points within the speicified X-range.

CalcUnwrap

Command Syntax: SR1.Displays.Graph(i).Trace(j).CalcUnwrap(*YToleranceUnit*, *YTolerance*)

Command Argument(s): *YToleranceUnit* as String
YTolerance as Double

Return Value: *CalcTraceID* as Integer

Example: `CalcTraceID = SR1.Displays.Graph(1).Trace(101).CalcUnwra`

Description: Returns the traceID of a new trace continaing the "unwrapped" version of the original phase curve. Causes an execution error if the original trace is not phase vs. frequency. The tolerance should be 180 deg, or pi radians.

2.4.15.1.2 Graph Cursor

Object:	**SR1.Displays.Graph(*i*).Cursor**
Object Argument(s):	*i* as Integer
Description:	Commands related to the graph cursor.

Enabled

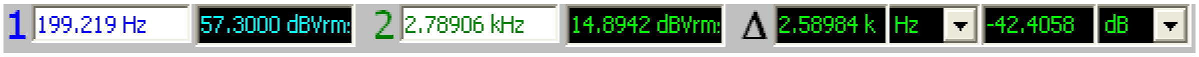
Command Syntax: SR1.Displays.Graph(i).Cursor.Enabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(2).Cursor.Enabled
SR1.Displays.Graph(2).Cursor.Enabled = False
```

Description: Enables or disables graph cursors.



1 199.219 Hz 57.3000 dBVrm: 2 2.78906 kHz 14.8942 dBVrm: Δ 2.58984 k Hz -42.4058 dB

x1

Command Syntax: SR1.Displays.Graph(i).Cursor.x1(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(2).Cursor.x1("Hz")
SR1.Displays.Graph(2).Cursor.x1("Hz") = 199.219
```

Description: Cursor 1 X-axis position.

x2

Command Syntax: SR1.Displays.Graph(i).Cursor.x2(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Graph(2).Cursor.x2("Hz")
SR1.Displays.Graph(2).Cursor.x2("Hz") = 2.78906
```

Description: Cursor 2 X-axis position.

y1Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.y1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Graph(2).Cursor.y1Rdg("dBVrms")
```

Description: Cursor 1 Y-value..

y2Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.y2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Graph(2).Cursor.y2Rdg("dbVrms")
```

Description: Cursor 2 Y-value.

dxCalc

Command Syntax: SR1.Displays.Graph(i).Cursor.dxCalc

Command Argument(s):

Property: CursDeltaCalc enumeration {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

Example:
```
val = SR1.Displays.Graph(2).Cursor.dxCalc
SR1.Displays.Graph(2).Cursor.dxCalc = cdDelta
```

Description: Method for calculating the X-axis difference.

dxRdg

Command Syntax: SR1.Displays.Graph(i).Cursor.dxRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Graph(2).Cursor.dxRdg("Hz")
```

Description: Cursor1/cursor2 X-axis difference.

GetDxUnit

Command Syntax: SR1.Displays.Graph(i).Cursor.GetDxUnit()

Command Argument(s): None

Return Value: *dxUnit* as String

Example:
```
dxUnit = SR1.Displays.Graph(2).Cursor.GetDxUnit()
```

Description: Units of the dXRdg value.

dyCalc

Command Syntax: SR1.Displays.Graph(i).Cursor.dyCalc

Command Argument(s):

Property: CursDeltaCalc enumeration {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

Example:
```
val = SR1.Displays.Graph(2).Cursor.dyCalc
SR1.Displays.Graph(2).Cursor.dyCalc = cdDelta
```

Description: Method used to calculate the Y-axis difference.

dyRdg

Command Syntax: SR1.Displays.Graph(i).Cursor.dyRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Graph(2).Cursor.dyRdg("DB")
```

Description: Delta Y value.

GetDyUnit

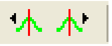
Command Syntax: SR1.Displays.Graph(i).Cursor.GetDyUnit()

Command Argument(s): None

Return Value: *dyUnit* as String

Example:
```
dyUnit = SR1.Displays.Graph(2).Cursor.GetDyUnit()
```

Description: Units of the dyRdg reading.



MoveToMax

Command Syntax: SR1.Displays.Graph(i).Cursor.MoveToMax(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example:
```
Call SR1.Displays.Graph(2).Cursor.MoveToMax(ciCursor1)
```

Description: Moves the specified cursor to the position corresponding to the maximum Y-value in the trace *within the displayed x-limits*.

MoveToMin

Command Syntax: SR1.Displays.Graph(i).Cursor.MoveToMin(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.MoveToMin(ciCursor2)`

Description: Moves the specified cursor to the position corresponding to the minimum Y-value in the trace *within the displayed x-limits*.

MoveToPeakL

Command Syntax: SR1.Displays.Graph(i).Cursor.MoveToPeakL(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.MoveToPeakL(ciCursor1)`

Description: Moves the specified cursor to the next peak left.

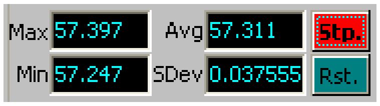
MoveToPeakR

Command Syntax: SR1.Displays.Graph(i).Cursor.MoveToPeakR(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.MoveToPeakR(ciCursor2)`

Description: Moves the specified cursor to the next peak right.



Max 57.397 Avg 57.311
Min 57.247 SDev 0.037555

Avg1Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Avg1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Avg1Rdg("")`

Description: Average value of cursor 1.

Avg2Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Avg2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Avg2Rdg("")`

Description: Average value of cursor 2. .

Max1Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Max1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Max1Rdg("")`

Description: Maximum value of cursor 1.

Max2Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Max2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Max2Rdg("")`

Description: Maximum value of cursor 2.

Min1Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Min1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Min1Rdg("")`

Description: Minimum value of cursor 1.

Min2Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Min2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Min2Rdg("")`

Description: Minimum value of cursor 2.

Sigma1Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Sigma1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Sigma1Rdg("")`

Description: Standard deviation of cursor 1.

Sigma2Rdg

Command Syntax: SR1.Displays.Graph(i).Cursor.Sigma2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.Sigma2Rdg("")`

Description: Standard deviation of cursor 2.

ResetStats

Command Syntax: SR1.Displays.Graph(i).Cursor.ResetStats(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.ResetStats(ciCursor1)`

Description: Resets the cursor statistics calculation for the specified cursor.

StartStats

Command Syntax: SR1.Displays.Graph(i).Cursor.StartStats(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.StartStats(ciCursor1)`

Description: Starts the cursor statistics calculation for the specified cursor.

StopStats

Command Syntax: SR1.Displays.Graph(i).Cursor.StopStats(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.StopStats(ciCursor1)`

Description: Stops cursor statistics calculations for the specified cursor.

ToggleStats

Command Syntax: SR1.Displays.Graph(i).Cursor.ToggleStats(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.Graph(2).Cursor.ToggleStats(ciCursor1)`

Description: Toggles the start/stopped state of cursor statistics calculation for the specified cursor.

☑ Ind. A1:FFT1:Power Spec A1:FFT1:Power Spec □ Lock X

Cursor1Trace

Command Syntax: SR1.Displays.Graph(i).Cursor.Cursor1Trace

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Displays.Graph(2).Cursor.Cursor1Trace
SR1.Displays.Graph(2).Cursor.Cursor1Trace = 1
```

Description: Returns an index into the trace list corresponding to the trace associated with cursor 1.

Cursor2Trace

Command Syntax: SR1.Displays.Graph(i).Cursor.Cursor2Trace

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Displays.Graph(2).Cursor.Cursor2Trace
SR1.Displays.Graph(2).Cursor.Cursor2Trace = 2
```

Description: Returns an index into the trace list corresponding to the trace associated with cursor 2.

Independent

Command Syntax: SR1.Displays.Graph(i).Cursor.Independent

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(2).Cursor.Independent
SR1.Displays.Graph(2).Cursor.Independent = False
```

Description: Independent cursor status (i.e. cursors allowed on different traces).

Xlock

Command Syntax: SR1.Displays.Graph(i).Cursor.Xlock

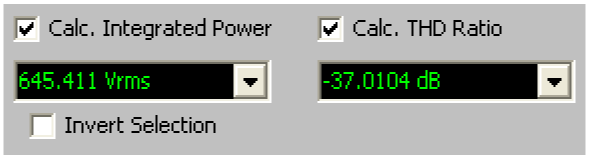
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(2).Cursor.Xlock
SR1.Displays.Graph(2).Cursor.Xlock = False
```

Description: When independent cursors are on, sets whether the two cursors X-values are locked together.



☑ Calc. Integrated Power

645.411 Vrms

☑ Calc. THD Ratio

-37.0104 dB

□ Invert Selection

CalcIntegratedPwr

Command Syntax: SR1.Displays.Graph(i).Cursor.CalcIntegratedPwr

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(2).Cursor.CalcIntegratedPwr
SR1.Displays.Graph(2).Cursor.CalcIntegratedPwr = False
```

Description: On/off status of integrated power calculation (applies to FFT spectra only). Power is calculated within the cursors.

CalcTHD

Command Syntax: SR1.Displays.Graph(i).Cursor.CalcTHD

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(2).Cursor.CalcTHD
SR1.Displays.Graph(2).Cursor.CalcTHD = False
```

Description: On/off status of cursor THD calculation. The left-most cursor marks the fundamental, while the right-most cursor marks the maximum frequency of interest.

IntegratedPwrInvertSel

Command Syntax: SR1.Displays.Graph(i).Cursor.IntegratedPwrInvertSel

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(2).Cursor.IntegratedPwrInvertSe
SR1.Displays.Graph(2).Cursor.IntegratedPwrInvertSel = Fa
```

Description: 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: SR1.Displays.Graph(i).Cursor.IntegratedPwrRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.IntegratedPwrRdg("Vrm`

Description: Returns the result of the integrated power calculation in the specified units.

THDRdg

Command Syntax: SR1.Displays.Graph(i).Cursor.THDRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

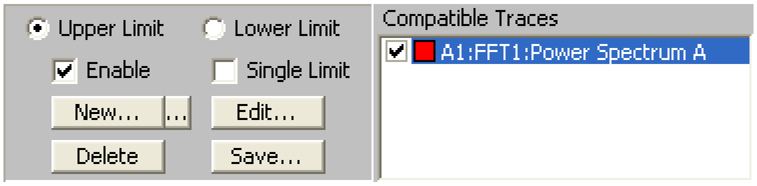
Property: Double (read-only)

Example: `val = SR1.Displays.Graph(2).Cursor.THDRdg("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.

2.4.15.1.3 Graph Limit

Object:	**SR1.Displays.Graph(i).GraphLimit**
Object Argument(s):	*i* as Integer
Description:	Commands related to graph limit-testing.



● Upper Limit ○ Lower Limit

☑ Enable □ Single Limit

Compatible Traces

☑ 🟥 A1:FFT1:Power Spectrum A

LimitFailedRdg

Command Syntax: SR1.Displays.Graph(i).GraphLimit.LimitFailedRdg

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(3).GraphLimit.LimitFailedRdg
```

Description: Returns True if any of the traces currently enabled for limit testing fail their limit tests. Otherwise returns false.

LoLimitEnabled

Command Syntax: SR1.Displays.Graph(i).GraphLimit.LoLimitEnabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(3).GraphLimit.LoLimitEnabled
SR1.Displays.Graph(3).GraphLimit.LoLimitEnabled = False
```

Description: Enabled/disabled status of the lower limit for the graph.

LoLimitExists

Command Syntax: SR1.Displays.Graph(i).GraphLimit.LoLimitExists

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:

```
val = SR1.Displays.Graph(3).GraphLimit.LoLimitExists
```

Description: Returns True if the graph has a configured lower limit.

LoLimitSingle

Command Syntax: SR1.Displays.Graph(i).GraphLimit.LoLimitSingle

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(3).GraphLimit.LoLimitSingle
SR1.Displays.Graph(3).GraphLimit.LoLimitSingle = False
```

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.

UpLimitEnabled

Command Syntax: SR1.Displays.Graph(i).GraphLimit.UpLimitEnabled

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(3).GraphLimit.UpLimitEnabled
SR1.Displays.Graph(3).GraphLimit.UpLimitEnabled = False
```

Description: Enabled/disabled status of the upper limit for the graph.

UpLimitExists

Command Syntax: SR1.Displays.Graph(i).GraphLimit.UpLimitExists

Command Argument(s):

Property: Boolean enumeration (read-only) {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(3).GraphLimit.UpLimitExists
```

Description: Returns True if an upper graph limit currently exists.

UpLimitSingle

Command Syntax: SR1.Displays.Graph(i).GraphLimit.UpLimitSingle

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.Graph(3).GraphLimit.UpLimitSingle
SR1.Displays.Graph(3).GraphLimit.UpLimitSingle = False
```

Description: Returns true if the upper limit is a single valued limit. Returns false if the upper limit is a multi-segment limit.

DelLimit

Command Syntax: SR1.Displays.Graph(i).GraphLimit.DelLimit(*LimitID*)

Command Argument(s): *LimitID* as limitId enumeration {lidUpLimit=0 | lidLoLimit=1}

Example:
```
Call SR1.Displays.Graph(3).GraphLimit.DelLimit(lidUpLimi
```

Description: Deletes the specified graph limit.

NewLimit

Command Syntax: SR1.Displays.Graph(i).GraphLimit.NewLimit(*LimitID*, *TraceID*)

Command Argument(s): *LimitID* as limitId enumeration {lidUpLimit=0 | lidLoLimit=1}
TraceID as Integer

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(3).GraphLimit.NewLimit(l`

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: SR1.Displays.Graph(i).GraphLimit.EditLimit(*LimitID*, *XData*, *XUnit*, *YData*, *YUnit*)

Command Argument(s): *LimitID* as limitId enumeration {lidUpLimit=0 | lidLoLimit=1}
XData as Array
XUnit as String
YData as Array
YUnit as String

Example: `Call SR1.Displays.Graph(3).GraphLimit.EditLimit(0, XData`

Description: Replaces the existing upper or lower limit with the limits specified in the XData and YData arrays.

SaveLimit

Command Syntax: SR1.Displays.Graph(i).GraphLimit.SaveLimit(*LimitID*, *FileName*)

Command Argument(s): *LimitID* as limitId enumeration {lidUpLimit=0 | lidLoLimit=1}
FileName as String

Example: `Call SR1.Displays.Graph(3).GraphLimit.SaveLimit(lidUpLim`

Description: Saves the upper or lower limit to the specified filename.

LoadLimit

Command Syntax: SR1.Displays.Graph(i).GraphLimit.LoadLimit(*LimitID*, *FileName*)

Command Argument(s): *LimitID* as limitId enumeration {lidUpLimit=0 | lidLoLimit=1}
FileName as String

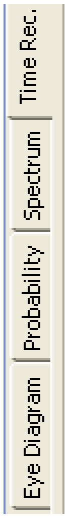
Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Graph(3).GraphLimit.LoadLimit(`

Description: Loads the limit contained in the specified file and returns the traceID of the limit trace.

2.4.15.2 Digitizer Display

Object:	**SR1.Displays.DigitizerDisplay(*i*)**
Object Argument(s):	*i* as Integer
Description:	Commands related to the Digitizer Display.



ActiveChart

Command Syntax: SR1.Displays.DigitizerDisplay(i).ActiveChart

Command Argument(s):

Property: ChartType enumeration {ddTimeRec=0 | ddSpectrum=1 | ddProbability=2 | ddEyeDiagram=3}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).ActiveChart
SR1.Displays.DigitizerDisplay(0).ActiveChart = ddTimeRec
```

Description: Active tab of the digitizer display.

Title

Command Syntax: SR1.Displays.DigitizerDisplay(i).Title

Command Argument(s):

Property: String

Example:

```
val = SR1.Displays.DigitizerDisplay(0).Title
SR1.Displays.DigitizerDisplay(0).Title = "My DUT"
```

Description: Returns the title of the digitizer display.

BackgroundColor

Command Syntax: SR1.Displays.DigitizerDisplay(i).BackgroundColor

Command Argument(s):

Property: backgroundColor enumeration {bcWhite=0 | bcBlack=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).BackgroundColor
SR1.Displays.DigitizerDisplay(0).BackgroundColor = bcWhi
```

Description: Background color of the display.

Save

Command Syntax: SR1.Displays.DigitizerDisplay(i).Save(*FileName*, *GraphData*)

Command Argument(s): *FileName* as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example:

```
Call SR1.Displays.DigitizerDisplay(0).Save("MyDigDisp.xm
```

Description: Saves the digitizer display to the named file with the specified data options.

AutoscaleOnAcquire

Command Syntax: SR1.Displays.DigitizerDisplay(i).AutoscaleOnAcquire

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).AutoscaleOnAcquir
SR1.Displays.DigitizerDisplay(0).AutoscaleOnAcquire = Fa
```

Description: Sets whether the displays will autoscale when a new digitizer record is acquired.

Clock 49.736 kHz

ClockRdg

Command Syntax: SR1.Displays.DigitizerDisplay(i).ClockRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Displays.DigitizerDisplay(0).ClockRdg("Hz")
```

Description: Returns the digital audio sampling clock frequency recovered from the digitizer record.

CursorEnabled

Command Syntax: SR1.Displays.DigitizerDisplay(i).CursorEnabled

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).CursorEnabled
SR1.Displays.DigitizerDisplay(0).CursorEnabled = False
```

Description: Returns the on/off state of the digitizer display cursor.

Maximize

Command Syntax: SR1.Displays.DigitizerDisplay(i).Maximize

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).Maximize
SR1.Displays.DigitizerDisplay(0).Maximize = False
```

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.

Online

Command Syntax: SR1.Displays.DigitizerDisplay(i).Online

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).Online
SR1.Displays.DigitizerDisplay(0).Online = False
```

Description: Online/offline status of the display.

Close

Command Syntax: SR1.Displays.DigitizerDisplay(i).Close()

Command Argument(s): None

Example:

```
Call SR1.Displays.DigitizerDisplay(0).Close()
```

Description: Closes the digitizer display form.

ExportData

Command Syntax: SR1.Displays.DigitizerDisplay(i).ExportData(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Displays.DigitizerDisplay(0).ExportData("MyData`

Description: Exports the data on the active page of the digitizer display to a text file with the specified filename.

ExportGraph

Command Syntax: SR1.Displays.DigitizerDisplay(i).ExportGraph(*Kind*, *FileName*)

Command Argument(s): *FileType* as FileKind enumeration {0=BitMap, 1=Enhanced MetaFile, 2=JPEG}
FileName as String

Example: `Call SR1.Displays.DigitizerDisplay(0).ExportGraph(2, "My`

Description: Exports the graph to the specified type of graphics file with the given filename.

Print

Command Syntax: SR1.Displays.DigitizerDisplay(i).Print()

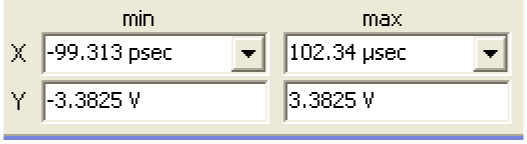
Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(0).Print()`

Description: Prints the digitizer display to the currently configured windows printer.

2.4.15.2.1 Digitizer Display Time Record

Object:	**SR1.Displays.DigitizerDisplay(*i*). TimeRecChart**
Object Argument(s):	None
Description:	Commands related to the Time Record tab of the Digitizer Display



	min	max
X	-99.313 psec	102.34 µsec
Y	-3.3825 V	3.3825 V

InputYmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.InputYmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Inpu
SR1.Displays.DigitizerDisplay(0).TimeRecChart.InputYmax(
```

Description: Maximum Y-axis value for the input trace.

InputYmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.InputYmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Inpu
SR1.Displays.DigitizerDisplay(0).TimeRecChart.InputYmin(
```

Description: Minimum Y-axis value for the input trace.

JitterYmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.JitterYmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Jitt
SR1.Displays.DigitizerDisplay(0).TimeRecChart.JitterYmax
```

Description: Maximum Y-axis value for the jitter trace.

JitterYmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.JitterYmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Jitt
SR1.Displays.DigitizerDisplay(0).TimeRecChart.JitterYmin
```

Description: Minimum Y-axis value for the jitter trace.

Xmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.Xmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Xmax
SR1.Displays.DigitizerDisplay(0).TimeRecChart.Xmax("SEC"
```

Description: Maximum X-axis value for the time record graph.

Xmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.Xmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Xmin
SR1.Displays.DigitizerDisplay(0).TimeRecChart.Xmin("S")
```

Description: Minimum X-axis value for the time record graph.

AutoScale

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.AutoScale()

Command Argument(s): None

Example:
```
Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.AutoS
```

Description: Autoscales the X and Y axes of the time record graph.

AutoScaleX

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.AutoScaleX()

Command Argument(s): None

Example:
```
Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.AutoS
```

Description: Autoscales the X axis of the time record graph.

AutoScaleY

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.AutoScaleY()

Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.AutoS`

Description: Autoscales the Y axis of the time record graph.

ShiftX

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.ShiftX(*Direction*)

Command Argument(s): *Direction* as Boolean {xLeft=0 | xRight=1}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.Shift`

Description: Shifts the X-axis of the time record graph left or right.

ShiftY

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.ShiftY(*Direction*)

Command Argument(s): *Direction* as Boolean {yDown=0 | yUp=1}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.Shift`

Description: Shifts the Y-axis of the time record graph up or down.

ZoomX

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.ZoomX(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.ZoomX`

Description: Zoom the X-axis in or out.

ZoomY

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.ZoomY(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.ZoomY`

Description: Zoom the Y-axis in or out.

☑ Input
☑ Jitter

SelectedTrace

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.SelectedTrace

Command Argument(s):

Property: TimeRecTraces enumeration {trInput=0 | trJitter=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Sele
SR1.Displays.DigitizerDisplay(0).TimeRecChart.SelectedTr
```

Description: Selected trace on the time record graph.

IsTraceChecked

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.IsTraceChecked(*Trace*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}

Return Value: *Checked* as Boolean {False=0 | True=1}

Example: `Checked = SR1.Displays.DigitizerDisplay(0).TimeRecChart.`

Description: Sets whether the specified trace (Input or Jitter) is checked.

TraceCheck

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.TraceCheck(*Trace*, *Check*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}
Check as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.Trace`

Description: Sets whether the specified trace (Input or Jitter) is checked.

SetColor

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.SetColor(*Trace*, *Color*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}
Color as Integer

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.SetCo`

Description: Sets the color of the specified trace.

SetWidth

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.SetWidth(*Trace*, *Width*)

Command Argument(s): Trace as *TimeRecTraces* enumeration {trInput=0 | trJitter=1}
Width as Integer

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.SetWi`

Description: Sets the width (in pixels) of the specified trace.

Data Commands

GetN

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.GetN(*Trace*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}

Return Value: *N* as Integer

Example: `N = SR1.Displays.DigitizerDisplay(0).TimeRecChart.GetN(t`

Description: Number of points in the specified trace.

GetXArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.GetXArray(*Trace*, *XUnit*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}
XUnit as String

Return Value: *XArray* as Array

Example: `XArray = SR1.Displays.DigitizerDisplay(0).TimeRecChart.G`

Description: Returns the X-axis array for the specified trace in the specified units.

GetYArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.GetYArray(*Trace*, *YUnit*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}
YUnit as String

Return Value: *YArray* as Array

Example: `YArray = SR1.Displays.DigitizerDisplay(0).TimeRecChart.G`

Description: Returns the Y-axis array for the specified trace in the specified units.

GetXY

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.GetXY(*Trace*, *Index*, *XUnit*, *YUnit*)

Command Argument(s): *Trace* as TimeRecTraces enumeration {trInput=0 | trJitter=1}
Index as Integer
XUnit as String
YUnit as String

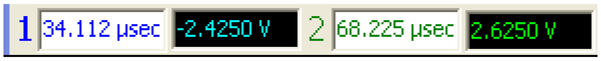
Return Value: *XYRdg* as Array

Example: `XYRdg = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Ge`

Description: Returns the X,Y pair corresponding to the Indexth point of the specified trace in the specified units.

2.4.15.2.1.1 Digitizer Display Cursor

Object:	**SR1.Displays.DigitizerDisplay(*i*). TimeRecChart.DigitizerCursor**
Object Argument(s):	None
Description:	Commands related to the cursor of the digitizer display. Note that each Chart (TimeRec,Chart, SpecChart, ProbChart, and EyeDiagram) has its own cursor. For brevity, only the cursor of the TimeRecChart is explained.



1 34.112 µsec -2.4250 V 2 68.225 µsec 2.6250 V

x1

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.x1(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi
SR1.Displays.DigitizerDisplay(0).TimeRecChart.DigitizerC
```

Description: Cursor 1 X-value.

x2

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.x2(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi
SR1.Displays.DigitizerDisplay(0).TimeRecChart.DigitizerC
```

Description: Cursor 2 X-value.

y1Rdg

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.y1Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi
```

Description: Returns the cursor 1 Y-Value.

y2Rdg

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.y2Rdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi`

Description: Returns the cursor 2 Y-Value.

Δ 34.112 µ sec 5.0500 V

dxCalc

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.dxCalc

Command Argument(s):

Property: CursDeltaCalc enumeration {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

Example:
```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi
SR1.Displays.DigitizerDisplay(0).TimeRecChart.DigitizerC
```

Description: Delta-X calculation method.

dxRdg

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.dxRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi`

Description: Returns the delta-X value.

dyCalc

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.dyCalc

Command Argument(s):

Property: CursDeltaCalc enumeration {cdDelta=0 | cdDeltaPct=1 | cdDeltaPPM=2 | cdDeltaHz=3 | cdRatio=4 | cdRatioPct=5 | cdRatioPPM=6 | cdRatiodB=7 | cdRatioDecade=8 | cdRatioOctave=9 | cdRatioCents=10}

Example:
```
val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi
SR1.Displays.DigitizerDisplay(0).TimeRecChart.DigitizerC
```

Description: Delta-Y calculation method..

dyRdg

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.dyRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digi`

Description: Delta-Y value.

GetDxUnit

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.GetDxUnit()

Command Argument(s): None

Return Value: *dxUnit* as String

Example: `dxUnit = SR1.Displays.DigitizerDisplay(0).TimeRecChart.D`

Description: Returns a string containing the units of the delta-X measurement.

GetDyUnit

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.GetDyUnit()

Command Argument(s): None

Return Value: *dyUnit* as String

Example: `dyUnit = SR1.Displays.DigitizerDisplay(0).TimeRecChart.D`

Description: Returns a string containing the units of the delta-Y measurement.

MoveToMax

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.MoveToMax(*CursorIndex*)

Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digit`

Description: Moves the specified cursor to the position corresponding to the maximum Y-value.

MoveToMin

Command Syntax: SR1.Displays.DigitizerDisplay(i).TimeRecChart.DigitizerCursor.MoveToMin(*CursorIndex*)

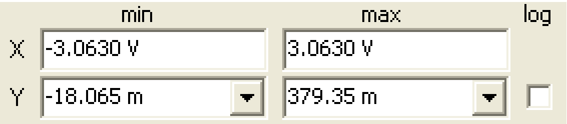
Command Argument(s): *CursorIndex* as CsrIdx enumeration {ciCursor1=0 | ciCursor2=1 | ciActiveCursor=2}

Example: `Call SR1.Displays.DigitizerDisplay(0).TimeRecChart.Digit`

Description: Moves the specified cursor to the position corresponding to the maximum Y-value.

2.4.15.2.2 Digitizer Display Probability

Object:	**SR1.Displays.DigitizerDisplay(*i*). ProbChart**
Object Argument(s):	None
Description:	Commands related to the Probability tab of the digitizer display.



	min	max	log
X	-3.0630 V	3.0630 V	
Y	-18.065 m	379.35 m	□

InputXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.InputXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.InputXm
SR1.Displays.DigitizerDisplay(1).ProbChart.InputXmax("V"
```

Description: Maximum X-axis value for the input trace.

InputXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.InputXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.InputXm
SR1.Displays.DigitizerDisplay(1).ProbChart.InputXmin("V"
```

Description: Minimum X-axis value for the input trace.

JitterXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.JitterXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.JitterX
SR1.Displays.DigitizerDisplay(1).ProbChart.JitterXmax("s
```

Description: Maximum X-axis value for the jitter trace.

JitterXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.JitterXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.JitterX
SR1.Displays.DigitizerDisplay(1).ProbChart.JitterXmin("s
```

Description: Minimum X-axis value for the jitter trace.

PulseRateXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.PulseRateXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.PulseRa
SR1.Displays.DigitizerDisplay(1).ProbChart.PulseRateXmax
```

Description: Maximum X-axis value for the pulse width trace.

PulseRateXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.PulseRateXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.PulseRa
SR1.Displays.DigitizerDisplay(1).ProbChart.PulseRateXmin
```

Description: Minimum X-axis value for the pulse width trace.

PulseWidthXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.PulseWidthXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.PulseWi
SR1.Displays.DigitizerDisplay(1).ProbChart.PulseWidthXma
```

Description: Maximum X-axis value for the pulse rate trace.

PulseWidthXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.PulseWidthXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.PulseWi
SR1.Displays.DigitizerDisplay(1).ProbChart.PulseWidthXmi
```

Description: Minimum X-axis value for the pulse rate trace.

Ylog

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.Ylog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.Ylog
SR1.Displays.DigitizerDisplay(1).ProbChart.Ylog = False
```

Description: Log/linear status of the Y-axis of the probability tab of the digitizer display.

Ymax

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.Ymax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.Ymax(""
SR1.Displays.DigitizerDisplay(1).ProbChart.Ymax("") = 0.
```

Description: Maximum Y-axis value for the probability graph.

Ymin

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.Ymin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.Ymin(""
SR1.Displays.DigitizerDisplay(1).ProbChart.Ymin("") = 0.
```

Description: Minimum Y-axis value for the probability graph.

AutoScale

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.AutoScale()

Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.AutoScal`

Description: Autoscales the X and Y axes of the probability graph.

AutoScaleX

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.AutoScaleX()

Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.AutoScal`

Description: Autoscales the X-axis of the probability graph.

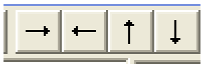
AutoScaleY

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.AutoScaleY()

Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.AutoScal`

Description: Autoscales the Y-axis of the probability graph.



ShiftX

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.ShiftX(*Direction*)

Command Argument(s): *Direction* as Boolean {xLeft=0 | xRight=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.ShiftX(x`

Description: Shifts the X-axis of the probability graph left or right.

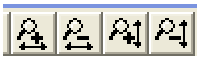
ShiftY

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.ShiftY(*Direction*)

Command Argument(s): *Direction* as Boolean {yDown=0 | yUp=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.ShiftY(y`

Description: Shifts the Y-axis of the probability graph left or right.



ZoomX

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.ZoomX(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.ZoomX(zm`

Description: Zooms the X-axis of the probability graph in or out.

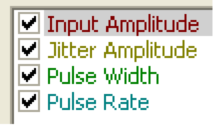
ZoomY

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.ZoomY(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.ZoomY(zm`

Description: Zooms the Y-axis of the probability graph in or out.



☑ Input Amplitude
☑ Jitter Amplitude
☑ Pulse Width
☑ Pulse Rate

SelectedTrace

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.SelectedTrace

Command Argument(s):

Property: ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

Example:
```
val = SR1.Displays.DigitizerDisplay(1).ProbChart.Selecte
SR1.Displays.DigitizerDisplay(1).ProbChart.SelectedTrace
```

Description: Selected trace of the probability graph.

IsTraceChecked

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.IsTraceChecked(*Trace*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

Return Value: *Checked* as Boolean {False=0 | True=1}

Example: `Checked = SR1.Displays.DigitizerDisplay(1).ProbChart.IsT`

Description: Queries whether the specified trace is checked (visible).

TraceCheck

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.TraceCheck(*Trace*, *Check*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
Check as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.TraceChe`

Description: Checks or unchecks the specified trace of the probability graph.

SetColor

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.SetColor(*Trace*, *Color*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
Color as Integer

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.SetColor`

Description: Sets the color of the specified trace of the probability graph.

SetWidth

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.SetWidth(*Trace*, *Width*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
Width as Integer

Example: `Call SR1.Displays.DigitizerDisplay(1).ProbChart.SetWidth`

Description: Sets the width (in pixels) of the specified trace.

Data Commands

GetN

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.GetN(*Trace*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}

Return Value: *N* as Integer

Example: `N = SR1.Displays.DigitizerDisplay(1).ProbChart.GetN(pInp`

Description: Returns the number of points in the specified trace.

GetXArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.GetXArray(*Trace*, *XUnit*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
XUnit as String

Return Value: *XArray* as Array

Example: `XArray = SR1.Displays.DigitizerDisplay(1).ProbChart.GetX`

Description: Returns the X-axis array corresponding to the specified trace.

GetYArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.GetYArray(*Trace*, *YUnit*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
YUnit as String

Return Value: *YArray* as Array

Example:

```
YArray = SR1.Displays.DigitizerDisplay(1).ProbChart.GetY
```

Description: Returns the Y-axis array corresponding to the specified trace.

GetXY

Command Syntax: SR1.Displays.DigitizerDisplay(i).ProbChart.GetXY(*Trace*, *Index*, *XUnit*, *YUnit*)

Command Argument(s): *Trace* as ProbabilityTraces enumeration {pInput=0 | pJitter=1 | pPulseWidth=2 | pPulseRate=3}
Index as Integer
XUnit as String
YUnit as String

Return Value: *XYRdg* as Array

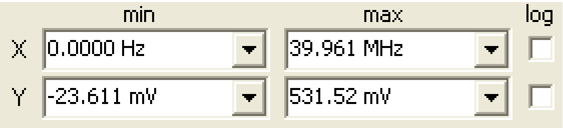
Example:

```
XYRdg = SR1.Displays.DigitizerDisplay(1).ProbChart.GetXY
```

Description: Returns X-Y pair corresponding to the Indexth point of the specified trace in the specified units.

2.4.15.2.3 Digitizer Display Spectrum

Object:	**SR1.Displays.DigitizerDisplay(*i*). SpecChart**
Object Argument(s):	None
Description:	Commands related to the spectrum tab on the digitizer display.



	min	max	log
X	0.0000 Hz	39.961 MHz	□
Y	-23.611 mV	531.52 mV	□

InputXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputXm
SR1.Displays.DigitizerDisplay(1).SpecChart.InputXmax("Hz
```

Description: Maximum X-axis value for the input spectrum.

InputXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputXm
SR1.Displays.DigitizerDisplay(1).SpecChart.InputXmin("")
```

Description: Minimum X-axis value for the input spectrum.

InputXlog

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputXlog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputXl
SR1.Displays.DigitizerDisplay(1).SpecChart.InputXlog = F
```

Description: Log/linear status of the spectrum X-axis.

InputYmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputYmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputYm
SR1.Displays.DigitizerDisplay(1).SpecChart.InputYmax("V"
```

Description: Maximum Y-axis value of the input spectrum.

InputYmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputYmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputYm
SR1.Displays.DigitizerDisplay(1).SpecChart.InputYmin("V"
```

Description: Minimum Y-axis value of the input spectrum.

InputYlog

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.InputYlog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.InputYl
SR1.Displays.DigitizerDisplay(1).SpecChart.InputYlog = F
```

Description: Log/linear status of the input spectrum Y-axis.

JitterXmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterXmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterX
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterXmax("H
```

Description: Maximum X-axis value of the jitter spectrum

JitterXmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterXmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterX
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterXmin("H
```

Description: Minimum X-axis value of the jitter spectrum

JitterXlog

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterXlog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterX
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterXlog =
```

Description: Log/linear status for the jitter spectrum X-axis.

JitterYmax

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterYmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterY
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterYmax("s
```

Description: Maximum Y-axis value for the jitter spectrum

JitterYmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterYmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterY
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterYmin("U
```

Description: Minimum Y-axis value for the jitter spectrum.

JitterYlog

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.JitterYlog

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.JitterY
SR1.Displays.DigitizerDisplay(1).SpecChart.JitterYlog =
```

Description: Log/linear status of the jitter spectrum Y-axis.

AutoScale

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.AutoScale()

Command Argument(s): None

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.AutoScal
```

Description: Autoscales the spectrum X anx Y axes.

AutoScaleX

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.AutoScaleX()

Command Argument(s): None

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.AutoScal
```

Description: Autoscales the spectrum X-axis.

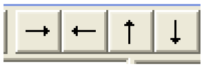
AutoScaleY

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.AutoScaleY()

Command Argument(s): None

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.AutoScal
```

Description: Autoscales the spectrum Y-axis.



ShiftX

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.ShiftX(*Direction*)

Command Argument(s): *Direction* as Boolean {xLeft=0 | xRight=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.ShiftX(x
```

Description: Shifts the spectrum X-axis left or right.

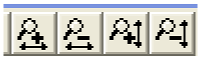
ShiftY

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.ShiftY(*Direction*)

Command Argument(s): *Direction* as Boolean {yDown=0 | yUp=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).SpecChart.ShiftY(y`

Description: Shifts the spectrum Y-axis up or down.



ZoomX

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.ZoomX(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).SpecChart.ZoomX(zm`

Description: Zooms the spectrum X-axis in or out.

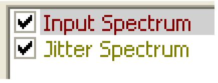
ZoomY

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.ZoomY(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example: `Call SR1.Displays.DigitizerDisplay(1).SpecChart.ZoomY(zm`

Description: Zooms the spectrum Y-axis in or out.



☑ Input Spectrum
☑ Jitter Spectrum

SelectedTrace

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.SelectedTrace

Command Argument(s):

Property: SpecTraces enumeration {sInput=0 | sJitter=1}

Example:
```
val = SR1.Displays.DigitizerDisplay(1).SpecChart.Selecte
SR1.Displays.DigitizerDisplay(1).SpecChart.SelectedTrace
```

Description: Selected trace of the spectrum graph.

IsTraceChecked

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.IsTraceChecked(*Trace*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}

Return Value: *Checked* as Boolean {False=0 | True=1}

Example: `Checked = SR1.Displays.DigitizerDisplay(1).SpecChart.IsT`

Description: Queries if the specified trace is checked (visible).

TraceCheck

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.TraceCheck(*Trace*, *Check*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
Check as Boolean {False=0 | True=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.TraceChe
```

Description: Checks or unchecks the specified trace of the spectrum graph.

SetColor

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.SetColor(*Trace*, *Color*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
Color as Integer

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.SetColor
```

Description: Sets the color of the specified trace.

SetWidth

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.SetWidth(*Trace*, *Width*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
Width as Integer

Example:
```
Call SR1.Displays.DigitizerDisplay(1).SpecChart.SetWidth
```

Description: Sets the width (in pixels) of the specified trace.

Data Commands

GetN

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.GetN(*Trace*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}

Return Value: *N* as Integer

Example:
```
N = SR1.Displays.DigitizerDisplay(1).SpecChart.GetN(sInp
```

Description: Returns the number of points in the specified trace of the spectrum graph..

GetXArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.GetXArray(*Trace*, *XUnit*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
XUnit as String

Return Value: *XArray* as Array

Example:
```
XArray = SR1.Displays.DigitizerDisplay(1).SpecChart.GetX
```

Description: Returns the X-data array for the specified spectrum trace in the specified units.

GetYArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.GetYArray(*Trace*, *YUnit*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
YUnit as String

Return Value: *YArray* as Array

Example:

```
YArray = SR1.Displays.DigitizerDisplay(1).SpecChart.GetY
```

Description: Returns the Y-data array for the specified spectrum trace in the specified units.

GetXY

Command Syntax: SR1.Displays.DigitizerDisplay(i).SpecChart.GetXY(*Trace*, *Index*, *XUnit*, *YUnit*)

Command Argument(s): *Trace* as SpecTraces enumeration {sInput=0 | sJitter=1}
Index as Integer
XUnit as String
YUnit as String

Return Value: *XYRdg* as Array

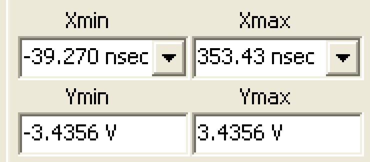
Example:

```
XYRdg = SR1.Displays.DigitizerDisplay(1).SpecChart.GetXY
```

Description: Returns X-Y pair corresponding to the Indexth point of the specified trace in the specified units.

2.4.15.2.4 Digitizer Display Eye Diagram

Object:	**SR1.Displays.DigitizerDisplay(*i*). EyeDiagram**
Object Argument(s):	None
Description:	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: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Xmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Xmax("
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Xmax("s") =
```

Description: Maximum X-axis value of the eye diagrm.

Xmin

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Xmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Xmin("
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Xmin("s") =
```

Description: Minimum X-axis value of the eye diagram.

Ymax

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Ymax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Ymax("
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Ymax("V") =
```

Description: Maximum Y-axis value of the eye diagram.

Ymin

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Ymin(*ValueUnit*)

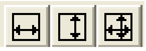
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Ymin("
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Ymin("V") =
```

Description: Minimum Y-axis value of the eye diagram.



AutoScale

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.AutoScale()

Command Argument(s): None

Example:

```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.AutoSca
```

Description: Autoscales the X and Y axes of the eye diagram.

AutoScaleX

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.AutoScaleX()

Command Argument(s): None

Example:

```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.AutoSca
```

Description: Autoscales the X-axis of the eye diagram.

AutoScaleY

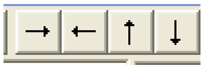
Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.AutoScaleY()

Command Argument(s): None

Example:

```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.AutoSca
```

Description: Autoscales the Y-axis of the eye diagram.



ShiftX

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.ShiftX(*Direction*)

Command Argument(s): *Direction* as Boolean {xLeft=0 | xRight=1}

Example:

```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.ShiftX(
```

Description: Shifts the X-axis of the eye diagram left or right..

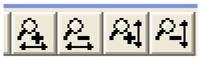
ShiftY

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.ShiftY(*Direction*)

Command Argument(s): *Direction* as Boolean {yDown=0 | yUp=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.ShiftY(
```

Description: Shifts the Y-axis of the eye diagram left or right.



ZoomX

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.ZoomX(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.ZoomX(z
```

Description: Zooms the X-axis of the eye diagram in or out.

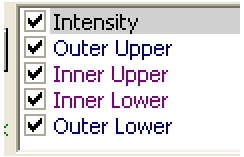
ZoomY

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.ZoomY(*Direction*)

Command Argument(s): *Direction* as Boolean {zmOut=0 | zmIn=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.ZoomY(z
```

Description: Zooms the Y-axis of the eye diagram in or out.



☑ Intensity
☑ Outer Upper
☑ Inner Upper
☑ Inner Lower
☑ Outer Lower

SelectedTrace

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.SelectedTrace

Command Argument(s):

Property: EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Select
SR1.Displays.DigitizerDisplay(2).EyeDiagram.SelectedTrac
```

Description: Returns the selected trace of the eye diagram.

IsTraceChecked

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.IsTraceChecked(*Trace*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

Return Value: *Checked* as Boolean {False=0 | True=1}

Example: `Checked = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Is`

Description: Checked (visible) status of the specified eye diagram trace.

TraceCheck

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.TraceCheck(*Trace*, *Check*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
Check as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.TraceCh`

Description: Checks or unchecks one of the eye diagram traces.

SetColor

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.SetColor(*Trace*, *Color*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
Color as Integer

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.SetColo`

Description: Sets the color of the specified trace.

SetWidth

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.SetWidth(*Trace*, *Width*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
Width as Integer

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.SetWidt`

Description: Sets the width (in pixels) of the specified trace.

Cursor1Trace

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Cursor1Trace

Command Argument(s):

Property: EyeDiagCursorTraces enumeration {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Cursor
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Cursor1Trace
```

Description: Eye diagram trace associated with cursor 1.

Cursor2Trace

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Cursor2Trace

Command Argument(s):

Property: EyeDiagCursorTraces enumeration {edcOuterUpper=0 | edcInnerUpper=1 | edcInnerLower=2 | edcOuterLower=3 | edcInUpLimit=4 | edcInLoLimit=5 | edcOutUpLimit=6 | edcOutLoLimit=7}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Cursor
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Cursor2Trace
```

Description: Eye diagram trace associated with cursor 2.

CursorXlock

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.CursorXlock

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Cursor
SR1.Displays.DigitizerDisplay(2).EyeDiagram.CursorXlock
```

Description: On/off status of the cursor X-lock.

MouseShowsZ

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.MouseShowsZ

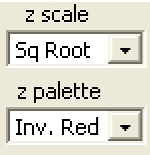
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.MouseS
SR1.Displays.DigitizerDisplay(2).EyeDiagram.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.



z scale
Sq Root
z palette
Inv. Red

Zpalette

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Zpalette

Command Argument(s):

Property: ZPaletteStyle enumeration {Bold=0 | Spectrum=1 | Grayscale=2 | RedHot=3 | GreenHot=4 | BlueHot=5 | InvGrayscale=6 | InvRedHot=7 | InvGreenHot=8 | InvBlueHot=9 | InvSpectrum=10}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Zpalet
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Zpalette = B
```

Description: Eye diagram color palette.

Zscale

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Zscale

Command Argument(s):

Property: ZScaling enumeration {zsLinear=0 | zsSqRoot=1 | zsLog=2}

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Zscale
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Zscale = zsL
```

Description: 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: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetN(*Trace*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration { edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}

Return Value: *N* as Integer

Example: `N = SR1.Displays.DigitizerDisplay(2).EyeDiagram.GetN(edO`

Description: Returns the number of points in the one-dimensional data array corresponding to the specified trace.

GetXY

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetXY(*Trace*, *Index*, *XUnit*, *YUnit*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
Index as Integer
XUnit as String
YUnit as String

Return Value: *XYRdg* as Array

Example: `XYRdg = SR1.Displays.DigitizerDisplay(2).EyeDiagram.GetX`

Description: Returns the X-Y pair corresponding to the Indexth point in the specified one-dimensional eye diagram array.

GetNxy

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetNxy()

Command Argument(s): None

Return Value: *Nxy* as Array

Example: `Nxy = SR1.Displays.DigitizerDisplay(2).EyeDiagram.GetNxy`

Description: Returns the size of the 2-dimensional array (rows, columns) corresponding to the eye diagram intensity measurement.

GetXYZ

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetXYZ(*XIndex*, *YIndex*, *XUnit*, *YUnit*, *ZUnit*)

Command Argument(s): *XIndex* as Integer
YIndex as Integer
XUnit as String
YUnit as String
ZUnit as String

Return Value: *XYZRdg* as Array

Example:
```
XYZRdg = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Get
```

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: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetXArray(*Trace*, *XUnit*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
XUnit as String

Return Value: *XArray* as Array

Example:
```
XArray = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Get
```

Description: Returns the X-axis array for the specified trace.

GetYArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetYArray(*Trace*, *YUnit*)

Command Argument(s): *Trace* as EyeDiagTraces enumeration {edIntensity=0 | edOuterUpper=1 | edInnerUpper=2 | edInnerLower=3 | edOuterLower=4 | edInUpLimit=5 | edInLoLimit=6 | edOutUpLimit=7 | edOutLoLimit=8}
YUnit as String

Return Value: *YArray* as Array

Example:
```
YArray = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Get
```

Description: Returns the Y-axis array for the specified trace.

GetZArray

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.GetZArray(*ZUnit*)

Command Argument(s): *ZUnit* as String

Return Value: *ZArray* as Array

Example:
```
ZArray = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Get
```

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.

2.4.15.2.4.1 Eye Diagram Limits

Object:	**SR1.Displays.DigitizerDisplay(*i*). EyeDiagram.Limit**
Object Argument (s):	None
Description:	Commands related to the eye diagram limits.

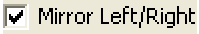
Exceeded

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Exceeded

Property: Integer (read-only)

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
```

Description: Returns 1 if the currently set eye limits are exceeded, 0 if not.



MirrorLR

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.MirrorLR

Command Argument(s):

Property: Integer {False=0 | True=1}

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.Mirror
```

Description: 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.

MirrorUD

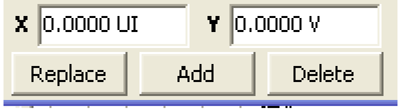
Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.MirrorUD

Command Argument(s):

Property: Integer {False=0 | True=1}

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.Mirror
```

Description: 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.


X 0.0000 UI Y 0.0000 V

AddPoint

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.AddPoint(*Trace*, *X*, *Y*)

Command Argument(s): *Trace* as LimitTraces enumeration {InUp=0 | InLo=1}
X as Double
Y as Double

Return Value: *Index* as Integer

Example: `Index = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limi`

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: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.ChangePoint(*Trace*, *Index*, *X*, *Y*)

Command Argument(s): *Trace* as LimitTraces enumeration {InUp=0 | InLo=1}
Index as Integer
X as Double
Y as Double

Return Value: *NewIndex* as Integer

Example: `NewIndex = SR1.Displays.DigitizerDisplay(2).EyeDiagram.L`

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: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.DelPoint(*Trace*, *Index*)

Command Argument(s): *Trace* as LimitTraces enumeration {InUp=0 | InLo=1}
Index as Integer

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.D`

Description: Deletes the point from the specified trace with the given index.

NumPoints

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.NumPoints(*Trace*)

Command Argument(s): *Trace* as LimitTraces enumeration {InUp=0 | InLo=1}

Return Value: *Count* as Integer

Example: `Count = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limi`

Description: Returns the number of points in the specified limit.

Point

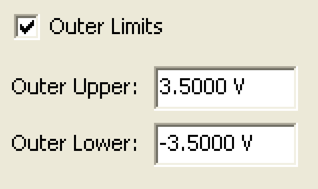
Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Point(*Trace*, *Index*)

Command Argument(s): *Trace* as LimitTraces enumeration {InUp=0 | InLo=1}
Index as Integer

Return Value: *Point* as Array

Example: `Point = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limi`

Description: Returns the X,Y pair corresponding to the Indexth point of the specified limit trace.



☑ Outer Limits

Outer Upper: 3.5000 V

Outer Lower: -3.5000 V

EnableOuter

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.EnableOuter(*On*)

Command Argument(s): *On* as Boolean {False=0 | True=1}

Example:
```
Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.E
```

Description: Enables or disables checking of the outer eye limits.

OutLoLimit

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.OutLoLimit(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.OutLoL
```

Description: Value of the outer lower limit.

OutUpLimit

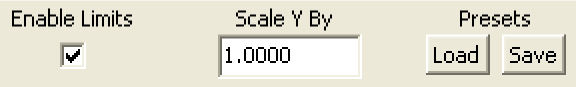
Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.OutUpLimit(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.OutUpL
```

Description: Value of the outer upper limit.



Enable Limits ☑

Scale Y By 1.0000

Enable

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Enable(*On*)

Command Argument(s): *On* as Boolean {False=0 | True=1}

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.E`

Description: Enables/dispables eye limit testing.

Yscale

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Yscale(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.
SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.Yscale
```

Description: Value of the limit Y-scale factor.

Load

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Load(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.L`

Description: Loads an eye limit configuration from the named file.

Save

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.Save(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.S`

Description: Saves the current eye limit configuration to the named file.

CloseForm

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.CloseForm()

Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.C`

Description: Opens the Eye Limits form

OpenForm

Command Syntax: SR1.Displays.DigitizerDisplay(i).EyeDiagram.Limit.OpenForm()

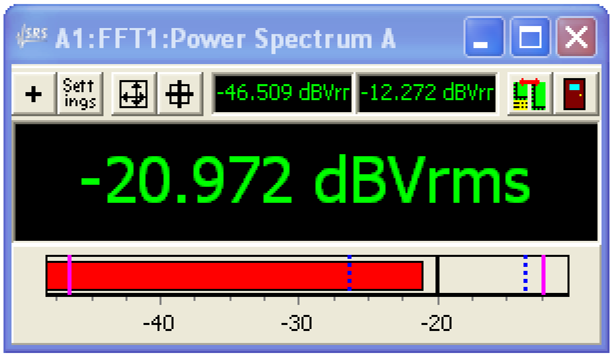
Command Argument(s): None

Example: `Call SR1.Displays.DigitizerDisplay(2).EyeDiagram.Limit.O`

Description: Closes the eye limits form.

2.4.15.3 BarChart

Object:	**SR1.Displays.Bar(*i*)**
Object Argument(s):	*i* as Integer
Description:	Commands related to the bar chart.



BarRdg

Command Syntax: SR1.Displays.Bar(i).BarRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Bar(0).BarRdg("dBVrms")`

Description: Returns the current reading of the bar chart.

AvgRdg

Command Syntax: SR1.Displays.Bar(i).AvgRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Bar(0).AvgRdg("dBVrms")`

Description: Returns the average value of the bar chart.

MaxRdg

Command Syntax: SR1.Displays.Bar(i).MaxRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example: `val = SR1.Displays.Bar(0).MaxRdg("dBVrms")`

Description: Returns the maximum bar chart reading.

MinRdg

Command Syntax: SR1.Displays.Bar(i).MinRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Bar(0).MinRdg("dBVrms")
```

Description: Returns the minimum bar chart reading.

SdevRdg

Command Syntax: SR1.Displays.Bar(i).SdevRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:
```
val = SR1.Displays.Bar(0).SdevRdg("Vrms")
```

Description: Standard deviation of the bar chart readings.

Reset

Command Syntax: SR1.Displays.Bar(i).Reset()

Command Argument(s): None

Example:
```
Call SR1.Displays.Bar(0).Reset()
```

Description: Resets the calculation of maximum, minimum, average and standard deviation.

Xmax

Command Syntax: SR1.Displays.Bar(i).Xmax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Bar(0).Xmax("Vp")
SR1.Displays.Bar(0).Xmax("Vp") = 1.90109E-05
```

Description: Maximum value of the bar display.

Xmin

Command Syntax: SR1.Displays.Bar(i).Xmin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.Displays.Bar(0).Xmin("Vp")
SR1.Displays.Bar(0).Xmin("Vp") = 2.92475E-06
```

Description: Minimum value of the bar display.

Maximize

Command Syntax: SR1.Displays.Bar(i).Maximize

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).Maximize
SR1.Displays.Bar(0).Maximize = False
```

Description: "Maximized" status of the bar chart. When maximized the bar disappears and the text area expands to fill the space.

NumOnly

Command Syntax: SR1.Displays.Bar(i).NumOnly

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).NumOnly
SR1.Displays.Bar(0).NumOnly = False
```

Description: Sets whether units will be displayed in the main text area of the bar chart.

SigFig

Command Syntax: SR1.Displays.Bar(i).SigFig

Command Argument(s):

Property: Integer

Example:

```
val = SR1.Displays.Bar(0).SigFig
SR1.Displays.Bar(0).SigFig = 5
```

Description: Number of significant figures used in the bar chart display.

Title

Command Syntax: SR1.Displays.Bar(i).Title

Command Argument(s):

Property: String

Example:

```
val = SR1.Displays.Bar(0).Title
SR1.Displays.Bar(0).Title = "DUT Thd+N"
```

Description: Sets the title string of the bar chart.

AddTrace

Command Syntax: SR1.Displays.Bar(i).AddTrace(*MeasID*)

Command Argument(s): *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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}

Return Value: *NewTraceID* as Integer

Example: `NewTraceID = SR1.Displays.Bar(0).AddTrace(msA0FFTspectru`

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: SR1.Displays.Bar(i).GetMeasID()

Command Argument(s): None

Return Value: *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.Displays.Bar(0).GetMeasID()`

Description: Returns the measurement ID currently associated with the bar chart.

GoOffline

Command Syntax: SR1.Displays.Bar(i).GoOffline()

Command Argument(s): None

Example: `Call SR1.Displays.Bar(0).GoOffline()`

Description: Takes the bar chart offline.

AutoScale

Command Syntax: SR1.Displays.Bar(i).AutoScale()

Command Argument(s): None

Example: `Call SR1.Displays.Bar(0).AutoScale()`

Description: Autoscales the bar portion of the display.

Close

Command Syntax: SR1.Displays.Bar(i).Close()

Command Argument(s): None

Example: `Call SR1.Displays.Bar(0).Close()`

Description: Closes the bar chart.

FormID

Command Syntax: SR1.Displays.Bar(i).FormID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.Displays.Bar(0).FormID()`

Description: Returns the formID of the chart allowing the size and position to be manipulated with the standard Form Commands.

Save

Command Syntax: SR1.Displays.Bar(i).Save(*FileName*, *GraphData*)

Command Argument(s): *FileName* as String
GraphData as SaveGraphDataOption enumeration {gdNeverSave=0 | gdAlwaysSave=1 | gdSaveOfflineOnly=2}

Example: `Call SR1.Displays.Bar(0).Save("MyBar.XML", gdNeverSave)`

Description: Saves the bar chart to the specified file with the specified data options.

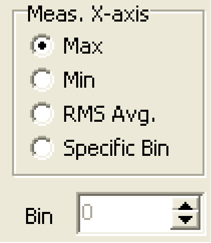
WriteMsg

Command Syntax: SR1.Displays.Bar(i).WriteMsg(*Message*)

Command Argument(s): *Message* as String

Example: `Call SR1.Displays.Bar(0).WriteMsg("Hello")`

Description: Writes a user message to the bar chart text area. This command only works when the bar chart is offline.



Meas. X-axis

● Max

○ Min

○ RMS Avg.

○ Specific Bin

Bin 0

Mode

Command Syntax: SR1.Displays.Bar(i).Mode

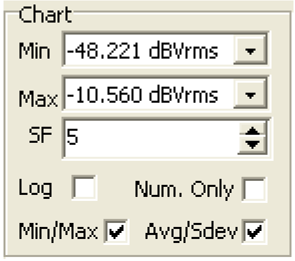
Command Argument(s):

Property: XAvgType enumeration {xaMax=0 | xaMin=1 | xaRMSAvg=2 | xaBin=3}

Example:

```
val = SR1.Displays.Bar(0).Mode
SR1.Displays.Bar(0).Mode = xaMax
```

Description: Selection mode for the bar chart when the associated measurement is a vector measurement.



Chart

Min -48.221 dBVrms

Max -10.560 dBVrms

SF 5

Log □ Num. Only □

Min/Max ☑ Avg/Sdev ☑

Readout

Command Syntax: SR1.Displays.Bar(i).Readout

Command Argument(s):

Property: StatsReadout enumeration {srMinMax=0 | srAvgSdev=1}

Example:

```
val = SR1.Displays.Bar(0).Readout
SR1.Displays.Bar(0).Readout = srMinMax
```

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: SR1.Displays.Bar(i).ShowAvgSdev

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).ShowAvgSdev
SR1.Displays.Bar(0).ShowAvgSdev = False
```

Description: Sets whether bars corresponding to the average/standard deviation are shown in the bar area.

ShowMinMax

Command Syntax: SR1.Displays.Bar(i).ShowMinMax

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).ShowMinMax
SR1.Displays.Bar(0).ShowMinMax = False
```

Description: Sets whether bars corresponding to the minimum and maximium are shown in the bar area.

Log

Command Syntax: SR1.Displays.Bar(i).Log

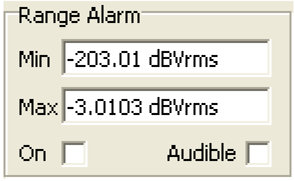
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).Log
SR1.Displays.Bar(0).Log = False
```

Description: Sets whether the bar display is linearly or logarithmically scaled.



Min -203.01 dBVrms

Max -3.0103 dBVrms

On ☐ Audible ☐

AlarmMax

Command Syntax: SR1.Displays.Bar(i).AlarmMax(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.Bar(0).AlarmMax("Vp")
SR1.Displays.Bar(0).AlarmMax("Vp") = 1.0
```

Description: Maximum value of the alarm range.

AlarmMin

Command Syntax: SR1.Displays.Bar(i).AlarmMin(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.Displays.Bar(0).AlarmMin("Vp")
SR1.Displays.Bar(0).AlarmMin("Vp") = -1.0
```

Description: Minimum value of the alarm range.

AlarmTone

Command Syntax: SR1.Displays.Bar(i).AlarmTone

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.Displays.Bar(0).AlarmTone
SR1.Displays.Bar(0).AlarmTone = False
```

Description: Sets whether an audible tone is used to signal and out-of-range condition.

RangeAlarm

Command Syntax: SR1.Displays.Bar(i).RangeAlarm

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

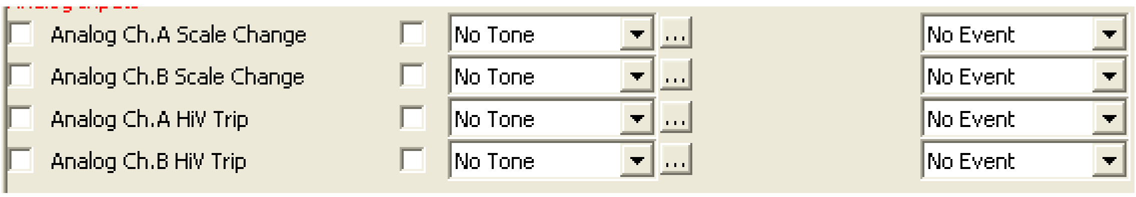
Example:

```
val = SR1.Displays.Bar(0).RangeAlarm
SR1.Displays.Bar(0).RangeAlarm = False
```

Description: Sets whether out-of-range conditions will trigger an alarm (color changes in the bar chart, remote events, and audible tone).

2.4.16 Event Manager

Object:	**SR1.EventMgr**
Object Argument(s):	None
Description:	Commands related to the Event Manager.



□	Analog Ch.A Scale Change	□	No Tone ▾	...	No Event ▾
□	Analog Ch.B Scale Change	□	No Tone ▾	...	No Event ▾
□	Analog Ch.A HiV Trip	□	No Tone ▾	...	No Event ▾
□	Analog Ch.B HiV Trip	□	No Tone ▾	...	No Event ▾

GetEnabled

Command Syntax: SR1.EventMgr.GetEnabled(*EventID*)

Command Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: *Enabled* as Boolean enumeration {False=0 | True=1}

Example: `Enabled = SR1.EventMgr.GetEnabled(eAInARngChg)`

Description: Enabled status for the specified event.

SetEnabled

Command Syntax: SR1.EventMgr.SetEnabled(*EventID*, *Enable*)

Command Argument(s): *EventID* as EventEnum enumeration {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 as Boolean {False=0 | True=1}

Example: `Call SR1.EventMgr.SetEnabled(eAInARngChg, False)`

Description: Sets the enabled status for the specified event.

GetLogging

Command Syntax: SR1.EventMgr.GetLogging(*EventID*)

Command Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: *Logged* as Boolean enumeration {False=0 | True=1}

Example: `Logged = SR1.EventMgr.GetLogging(eAInARngChg)`

Description: File logging status for the specified event.

SetLogging

Command Syntax: SR1.EventMgr.SetLogging(*EventID*, *Log*)

Command Argument(s): *EventID* as EventEnum enumeration {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 as Boolean enumeration {False=0 | True=1}

Example: `Call SR1.EventMgr.SetLogging(eAInARngChg, False)`

Description: Sets the file logging status for the specified event.

GetTone

Command Syntax: SR1.EventMgr.GetTone(*EventID*)

Command Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: *Tone* as ToneEnum enumeration {tnNoTone=0 | tnTone1=1 | tnTone2=2 | tnTone3=3 | tnTone4=4 | tnTone5=5}

Example: `Tone = SR1.EventMgr.GetTone(eAInARngChg)`

Description: Audio tone associated with the specified event.

SetTone

Command Syntax: SR1.EventMgr.SetTone(*EventID*, *Tone*)

Command Argument(s): *EventID* as EventEnum enumeration {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}
Tone as ToneEnum enumeration {tnNoTone=0 | tnTone1=1 | tnTone2=2 | tnTone3=3 | tnTone4=4 | tnTone5=5}

Example: `Call SR1.EventMgr.SetTone(eAInARngChg, tnNoTone)`

Description: Sets the audio tone associated with the specified event.

GetScript

Command Syntax: SR1.EventMgr.GetScript(*EventID*)

Command Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: *ScriptFile* as String

Example: `ScriptFile = SR1.EventMgr.GetScript(eAInARngChg)`

Description: Script file associated with the specified event.

SetScript

Command Syntax: SR1.EventMgr.SetScript(*EventID*, *ScriptFile*)

Command Argument(s): *EventID* as EventEnum enumeration {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 as String

Example: `Call SR1.EventMgr.SetScript(eAInARngChg, "MyScript.vbs")`

Description: Sets the script file associated with the specified event.

GetComEvent

Command Syntax: SR1.EventMgr.GetComEvent(*EventID*)

Command Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: *ComEvent* as ComEvents enumeration {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: `ComEvent = SR1.EventMgr.GetComEvent(eAInARngChg)`

Description: COM event associated with the specified event.

SetComEvent

Command Syntax: SR1.EventMgr.SetComEvent(*EventID*, *ComEvent*)

Command Argument(s): *EventID* as EventEnum enumeration {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 as ComEvents enumeration {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: `Call SR1.EventMgr.SetComEvent(eAInARngChg, ceEvent2)`

Description: Sets the COM event associated with the specified event.

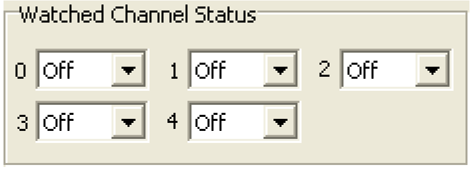
FireUserEvent

Command Syntax: SR1.EventMgr.FireUserEvent()

Command Argument(s): None

Example: `Call SR1.EventMgr.FireUserEvent()`

Description: Fires the COM "User Event".



Watched Channel Status

0 Off 1 Off 2 Off

3 Off 4 Off

GetWatchedChStat

Command Syntax: SR1.EventMgr.GetWatchedChStat(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Byte* as Integer

Example: `Byte = SR1.EventMgr.GetWatchedChStat(3)`

Description: Returns the channel status byte watched by the Indexth (0-4) channel status watcher.

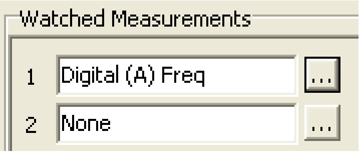
SetWatchedChStat

Command Syntax: SR1.EventMgr.SetWatchedChStat(*Index*, *Byte*)

Command Argument(s): *Index* as Integer
Byte as Integer

Example: `Call SR1.EventMgr.SetWatchedChStat(2, 4)`

Description: Sets the channel status byte watched by the Indexth (0-4) channel status watcher.



Watched Measurements

1 Digital (A) Freq

2 None

GetWatchedMeas

Command Syntax: SR1.EventMgr.GetWatchedMeas(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *MeasID* as MeasID enumeration {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 | 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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `MeasID = SR1.EventMgr.GetWatchedMeas(2)`

Description: Returns the id of the measurement watched by the Indexth (0-4) measurement watcher.

SetWatchedMeas

Command Syntax: SR1.EventMgr.SetWatchedMeas(*Index*, *MeasID*)

Command Argument(s): *Index* as Integer
MeasID as MeasID enumeration {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 |
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 | msA1MThdBinsA=2206 | msA1MThdBinsB=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: `Call SR1.EventMgr.SetWatchedMeas(2, msA1MThighestToneB)`

Description: Sets the id of the measurement watched by the Indexth (0-4) measurement watcher.

SetLogFile

Command Syntax: SR1.EventMgr.SetLogFile(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.EventMgr.SetLogFile("MyLogFile.txt")`

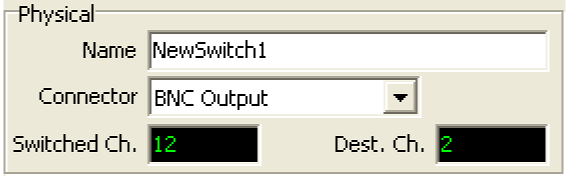
Description: Sets the log file associated with the Event Manager.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.17 Switcher Configuration

Object:	**SR1.Switcher**
Object Argument(s):	None
Description:	Commands related to controlling the switcher network.



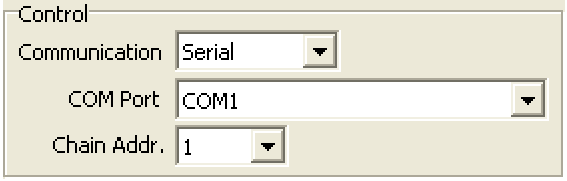
Physical

Name: NewSwitch1

Connector: BNC Output

Switched Ch.: 12

Dest. Ch.: 2



Control

Communication: Serial

COM Port: COM1

Chain Addr.: 1

AddNewSwitch

Command Syntax: SR1.Switcher.AddNewSwitch(*Name*, *SwitchType*, *ControlType*, *Address*, *ChainID*, *Order*)

Command Argument(s): *Name* as String
SwitchType as SwitchType enumeration {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}
ControlType as ControlType enumeration {ctSerial=0 | ctTCPIP=1}
Address as String
ChainID as Integer
Order as Integer

Example:

```
Call SR1.Switcher.AddNewSwitch("Switch1", 0, ctTCPIP, "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 thew newly added switch should be placed relative to existing switches. This can affect the logical port addresses for the switches.

ChangeDaisyChainAddress

Command Syntax: SR1.Switcher.ChangeDaisyChainAddress(*Type*, *Address*, *NewType*, *NewAddress*)

Command Argument(s): *Type* as ControlType enumeration {ctSerial=0 | ctTCPIP=1}
Address as String
NewType as ControlType enumeration {ctSerial=0 | ctTCPIP=1}
NewAddress as String

Example:

```
Call SR1.Switcher.ChangeDaisyChainAddress(ctTCPIP, "192.
```

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: SR1.Switcher.ChangeSwitchChainID(*Name*, *NewChainID*)

Command Argument(s): *Name* as String
NewChainID as Integer

Example: `Call SR1.Switcher.ChangeSwitchChainID("MySwitch", 12)`

Description: Changes the ChainID for the switch with the specified name to the value specified.

ChangeSwitchDaisyChain

Command Syntax: SR1.Switcher.ChangeSwitchDaisyChain(*Name*, *Type*, *Address*)

Command Argument(s): *Name* as String
Type as ControlType enumeration {ctSerial=0 | ctTCPIP=1}
Address as String

Example: `Call SR1.Switcher.ChangeSwitchDaisyChain("MySwitch", ctS`

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: SR1.Switcher.ChangeSwitchName(*Name*, *NewName*)

Command Argument(s): *Name* as String
NewName as String

Example: `Call SR1.Switcher.ChangeSwitchName("MySwitch", "YourSwit`

Description: Changes the name of the specified switch to the new value.

Network

Order 0

ChangeSwitchOrder

Command Syntax: SR1.Switcher.ChangeSwitchOrder(*Name*, *NewOrder*)

Command Argument(s): *Name* as String
NewOrder as Integer

Example: `Call SR1.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: SR1.Switcher.DeleteSwitch(*Name*)

Command Argument(s): *Name* as String

Example: `Call SR1.Switcher.DeleteSwitch("MySwitch")`

Description: Deletes the switch with the specified name.

GetChainNumSwitches

Command Syntax: SR1.Switcher.GetChainNumSwitches(*ControlType*)

Command Argument(s): *ControlType* as ControlType enumeration {ctSerial=0 | ctTCPIP=1}

Return Value: *N* as Integer

Example: `N = SR1.Switcher.GetChainNumSwitches(ctSerial)`

Description: Returns the number of switches with the specified control method.

GetChainSwitchName

Command Syntax: SR1.Switcher.GetChainSwitchName(*ControlType*, *Index*)

Command Argument(s): *ControlType* as ControlType enumeration {ctSerial=0 | ctTCPIP=1}
Index as Integer

Return Value: *Name* as String

Example: `Name = SR1.Switcher.GetChainSwitchName(ctSerial, 0)`

Description: Returns the name of the Indexth switch with the specified control method.

GetNetNumSwitches

Command Syntax: SR1.Switcher.GetNetNumSwitches(*SwitchType*)

Command Argument(s): *SwitchType* as SwitchType enumeration {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}

Return Value: *N* as Integer

Example: `N = SR1.Switcher.GetNetNumSwitches(stBNCOutput)`

Description: Returns the number of switches in the specified network (BNC In/Out, XLR In/Out).

GetNetSwitchName

Command Syntax: SR1.Switcher.GetNetSwitchName(*SwitchType*, *Index*)

Command Argument(s): *SwitchType* as SwitchType enumeration {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}
Index as Integer

Return Value: *Name* as String

Example: `Name = SR1.Switcher.GetNetSwitchName(stBNCOutput, 0)`

Description: Returns the name of the Indexth switch in the specified network (BNC In/Out, XLR In/Out).

IdentifySwitch

Command Syntax: SR1.Switcher.IdentifySwitch(*Name*)

Command Argument(s): *Name* as String

Example: `Call SR1.Switcher.IdentifySwitch("MySwitch")`

Description: Makes audible relay-clicks on the specified switch.

IsNameInUse

Command Syntax: SR1.Switcher.IsNameInUse(*Name*)

Command Argument(s): *Name* as String

Return Value: *InUse* as Integer

Example: `InUse = SR1.Switcher.IsNameInUse("MySwitch")`

Description: Returns 1 if the specified name is already in use as a switch name, otherwise returns 0.

MakeSwitch

Command Syntax: SR1.Switcher.MakeSwitch(*Name*, *PhysCh*, *State*)

Command Argument(s): *Name* as String
PhysCh as Integer
State as SwitchState enumeration {ssNone=-1 | ssA=0 | ssB=1}

Example: `Call SR1.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: SR1.Switcher.MakeSwitchEnum(*SwitchType*, *ChEnum*, *State*)

Command Argument(s): *SwitchType* as SwitchType enumeration {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}
ChEnum as Integer
State as SwitchState enumeration {ssNone=-1 | ssA=0 | ssB=1}

Example: `Call SR1.Switcher.MakeSwitchEnum(stBNCOutput, 22, ssNone`

Description: Connects the logical channel of the specified network to either A,B, or nothing (disconnected).

ResetNetwork

Command Syntax: SR1.Switcher.ResetNetwork(*SwitchType*)

Command Argument(s): *SwitchType* as SwitchType enumeration {stBNCOutput=0 | stBNCInput=1 | stXLROutput=2 | stXLRInput=3}

Example: `Call SR1.Switcher.ResetNetwork(stBNCOutput)`

Description: Resets (disconnects all inputs or outputs) all the switches in the specified network.

ResetSwitch

Command Syntax: SR1.Switcher.ResetSwitch(*Name*)

Command Argument(s): *Name* as String

Example: `Call SR1.Switcher.ResetSwitch("MySwitch")`

Description: Resets (disconnects all inputs or outputs) the named switch.

TestSwitchComms

Command Syntax: SR1.Switcher.TestSwitchComms(*Name*)

Command Argument(s): *Name* as String

Return Value: *Pass* as Boolean enumeration {False=0 | True=1}

Example: `Pass = SR1.Switcher.TestSwitchComms("MySwitch")`

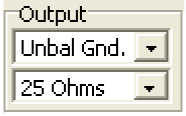
Description: Returns the status of a communication test with the named switch.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.18 Analog Generator

Object:	**SR1.AnlgGen**
Object Argument(s):	None
Description:	Commands related to the analog generator.



Output

Unbal Gnd.

25 Ohms

ConnectorConfig

Command Syntax: SR1.AnlgGen.ConnectorConfig

Command Argument(s):

Property: AnlgGenOutConfigType enumeration {aoUnbalGnd=0 | aoUnbalFloat=1 | aoBalGnd=2 | aoBalFloat=3 | aoBalCommon=4}

Example:

```
val = SR1.AnlgGen.ConnectorConfig
SR1.AnlgGen.ConnectorConfig = aoUnbalGnd
```

Description: Analog generator connector configuration.

Zout

Command Syntax: SR1.AnlgGen.Zout

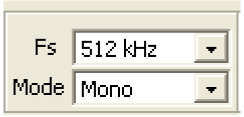
Command Argument(s):

Property: AnlgGenZout enumeration {aozBal50Un25=0 | aozBal150Un75=1 | aoz600=2}

Example:

```
val = SR1.AnlgGen.Zout
SR1.AnlgGen.Zout = aozBal50Un25
```

Description: Analog generator output impedance.



Fs 512 kHz

Mode Mono

Mono

Command Syntax: SR1.AnlgGen.Mono

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.Mono
SR1.AnlgGen.Mono = False
```

Description: Mono/stereo mode of the analog generator.

SampleRate

Command Syntax: SR1.AnlgGen.SampleRate

Command Argument(s):

Property: AnlgGenFs enumeration {agHz512k=0 | agHz128k=1 | agHz64k=2 | agOSR=3 | agISR=4}

Example:

```
val = SR1.AnlgGen.SampleRate
SR1.AnlgGen.SampleRate = agHz512k
```

Description: Sample rate mode of the analog generator.

SampleRateRdg

Command Syntax: SR1.AnlgGen.SampleRateRdg(*ValueUnit*)

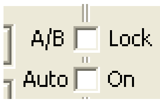
Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.AnlgGen.SampleRateRdg("Hz")
```

Description: Effective sample rate of the analog generator.



ABLock

Command Syntax: SR1.AnlgGen.ABLock

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.ABLock
SR1.AnlgGen.ABLock = False
```

Description: 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: SR1.AnlgGen.AutoOnOff

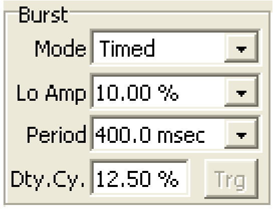
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AutoOnOff
SR1.AnlgGen.AutoOnOff = False
```

Description: 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.



Burst

Mode Timed

Lo Amp 10.00 %

Period 400.0 msec

Dty.Cy. 12.50 %

BurstDutyCycle

Command Syntax: SR1.AnlgGen.BurstDutyCycle(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.BurstDutyCycle("%")
SR1.AnlgGen.BurstDutyCycle("%") = 12.5
```

Description: Fraction of the burst period that the generator is at its high amplitude.

BurstLoAmp

Command Syntax: SR1.AnlgGen.BurstLoAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.BurstLoAmp("%")
SR1.AnlgGen.BurstLoAmp("%") = 10
```

Description: Burst lo amplitude (as a fraction of the high amplitude).

BurstMode

Command Syntax: SR1.AnlgGen.BurstMode

Command Argument(s):

Property: BurstType enumeration {bmNone=0 | bmTimed=1 | bmGatedHi=2 | bmGatedLo=3 | bmShaped=4 | bmTriggered=5}

Example:

```
val = SR1.AnlgGen.BurstMode
SR1.AnlgGen.BurstMode = bmNone
```

Description: Generator burst mode.

BurstPeriod

Command Syntax: SR1.AnlgGen.BurstPeriod(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.BurstPeriod("")
SR1.AnlgGen.BurstPeriod("") = Value
```

Description: Burst period.

BurstExecuteTrigger

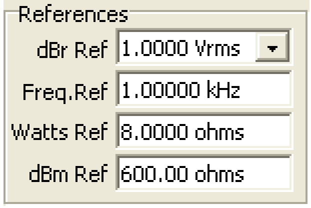
Command Syntax: SR1.AnlgGen.BurstExecuteTrigger()

Command Argument(s): None

Example:

```
Call SR1.AnlgGen.BurstExecuteTrigger()
```

Description: Sends a software burst trigger.



References

dBr Ref 1.0000 Vrms

Freq.Ref 1.00000 kHz

Watts Ref 8.0000 ohms

dBm Ref 600.00 ohms

dBmZref

Command Syntax: SR1.AnlgGen.dBmZref(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.dBmZref("OHMS")
SR1.AnlgGen.dBmZref("OHMS") = 600
```

Description: Reference impedance used for dBm calculations.

dBrRef

Command Syntax: SR1.AnlgGen.dBrRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.dBrRef("VRMS")
SR1.AnlgGen.dBrRef("VRMS") = 1
```

Description: Reference value used for dBr units.

FreqRef

Command Syntax: SR1.AnlgGen.FreqRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.FreqRef("Hz")
SR1.AnlgGen.FreqRef("Hz") = 1000
```

Description: Frequency used for relative frequency units.

WattsZref

Command Syntax: SR1.AnlgGen.WattsZref(*ValueUnit*)

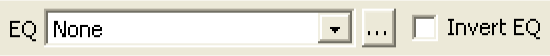
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.WattsZref("OHMS")
SR1.AnlgGen.WattsZref("OHMS") = 8
```

Description: Impedance used for analog generator Watts calculation.



EQ None

□ Invert EQ

SetEqFile

Command Syntax: SR1.AnlgGen.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.Alyzr(i).IMD.SetEqFile("MyEqFile.EQ")
```

Description: Sets the generator EQ file.

InvertEq

Command Syntax: SR1.AnlgGen.InvertEq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.InvertEq
SR1.AnlgGen.InvertEq = False
```

Description: Inverted/non-inverted status of the analog generator EQ file.

Form Commands:

OpenForm

OpenFormwID

CloseForm

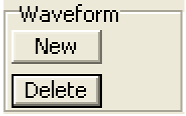
CloseForms

FormCount

FormID

2.4.18.1 Analog Generator Channel

Object:	**SR1.AnlgGen.AGenCh[A/B]**
Object Argument(s):	*ch* as A or B
Description:	Commands related to one channel of the analog generator.



AddWaveform

Command Syntax: SR1.AnlgGen.AGenCh[A/B].AddWaveform(*Type*)

Command Argument(s): *Type* as AnlgWaveformType enumeration {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}

Return Value: *ChanID* as Integer

Example: `ChanID = SR1.AnlgGen.AGenChA.AddWaveform(awfSine)`

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: SR1.AnlgGen.AGenChA.aSine(ChanID)

ClearWaveforms

Command Syntax: SR1.AnlgGen.AGenCh[A/B].ClearWaveforms()

Command Argument(s): None

Example: `Call SR1.AnlgGen.AGenChA.ClearWaveforms()`

Description: Clears all waveforms from the generator channel.

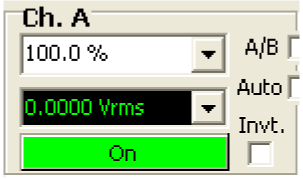
DeleteWaveform

Command Syntax: SR1.AnlgGen.AGenCh[A/B].DeleteWaveform(*ChanID*)

Command Argument(s): *ChanID* as Integer

Example: `Call SR1.AnlgGen.AGenChA.DeleteWaveform(3)`

Description: Clears the waveform with the specified ID.



Gain

Command Syntax: SR1.AnlgGen.AGenCh[A/B].Gain(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.Gain("%")
SR1.AnlgGen.AGenChB.Gain("%") = 100
```

Description: Overall gain of the generator channel.

Invert

Command Syntax: SR1.AnlgGen.AGenCh[A/B].Invert

Command Argument(s):

Property: Integer

Example:

```
val = SR1.AnlgGen.AGenChA.Invert
SR1.AnlgGen.AGenChB.Invert = Value
```

Description: Invert status for the generator channel.

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.On
SR1.AnlgGen.AGenChB.On = False
```

Description: On/off status for the generator channel.

TotalAmpRdg

Command Syntax: SR1.AnlgGen.AGenCh[A/B].TotalAmpRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.AnlgGen.AGenChA.TotalAmpRdg("Vrms")
```

Description: 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: SR1.AnlgGen.AGenCh[A/B].SignalCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.AnlgGen.AGenChB.SignalCount()`

Description: Total number of waveforms active on the current channel.

SignalID

Command Syntax: SR1.AnlgGen.AGenCh[A/B].SignalID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *ID* as Integer

Example: `ID = SR1.AnlgGen.AGenChA.SignalID(0)`

Description: ID of the Indexth waveform.

SignalType

Command Syntax: SR1.AnlgGen.AGenCh[A/B].SignalType(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Type* as AnlgWaveformType enumeration {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: `Type = SR1.AnlgGen.AGenChA.SignalType(2)`

Description: Waveform type of the Indexth waveform.

FreqRdg

Command Syntax: SR1.AnlgGen.AGenCh[A/B].FreqRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

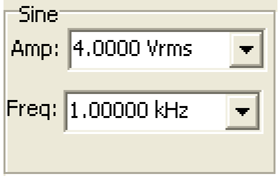
Property: Double (read-only)

Example: `val = SR1.AnlgGen.AGenChA.FreqRdg("Hz")`

Description: Frequency of the first waveform with a defined frequency (such as sine).

2.4.18.1.1 Sine

Object:	**SR1.AnlgGen.AGenCh[A/B].aSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator sine signal.



Sine
Amp: 4.0000 Vrms
Freq: 1.00000 kHz

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSine(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aSine(0).Amp("Vrms") = 4.0
```

Description: Sine amplitude.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSine(i).Freq(*ValueUnit*)

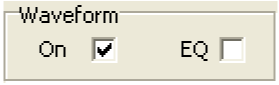
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSine(0).Freq("Hz")
SR1.AnlgGen.AGenChB.aSine(0).Freq("Hz") = 1000
```

Description: Sine frequency.



Waveform
On ☑ EQ □

Eq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChB.aSine(0).Eq
SR1.AnlgGen.AGenChA.aSine(0).Eq = False
```

Description: Enables or disables EQ for sine amplitude. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq.

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

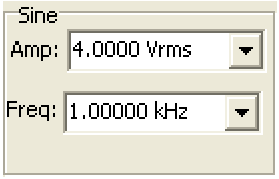
Example:

```
val = SR1.AnlgGen.AGenChA.aSine(0).On
SR1.AnlgGen.AGenChB.aSine(0).On = False
```

Description: On/off status of the sine waveform.

2.4.18.1.2 Lo Distortion Sine

Object:	**SR1.AnlgGen.AGenCh[A/B].aLoDistSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator low-distortion sine signal.



Sine

Amp: 4.0000 Vrms

Freq: 1.00000 kHz

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLoDistSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aLoDistSine(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aLoDistSine(0).Amp("Vrms") = 4.0
```

Description: Amplitude of the low-distortion sine.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLoDistSine(i).Freq(*ValueUnit*)

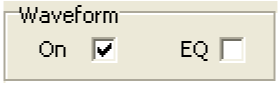
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aLoDistSine(0).Freq("Hz")
SR1.AnlgGen.AGenChA.aLoDistSine(0).Freq("Hz") = 1000
```

Description: Frequency of the low-distortion sine signal.



Waveform

On ☑ EQ □

Eq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLoDistSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.aLoDistSine(0).Eq
SR1.AnlgGen.AGenChA.aLoDistSine(0).Eq = False
```

Description: Sets whether the analog generator EQ file is applied to the low-distortion sine signal. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLoDistSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

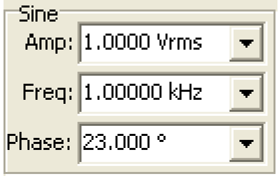
Example:

```
val = SR1.AnlgGen.AGenChA.aLoDistSine(0).On
SR1.AnlgGen.AGenChA.aLoDistSine(0).On = False
```

Description: Waveform on/off status.

2.4.18.1.3 Phased Sine

Object:	**SR1.AnlgGen.AGenCh[A/B]. aPhasedSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator phased sine waveform.


Sine
Amp: 1.0000 Vrms
Freq: 1.00000 kHz
Phase: 23.000 °

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPhasedSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aPhaseSine(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aPhaseSine(0).Amp("Vrms") = 1.0
```

Description: Amplitude of the phased sine waveform.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPhasedSine(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aPhaseSine(0).Freq("Hz")
SR1.AnlgGen.AGenChB.aPhaseSine(0).Freq("Hz") = 10000
```

Description: Phased sine frequency.

Phase

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPhasedSine(i).Phase(*ValueUnit*)

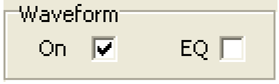
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aPhasedSine(0).Phase("°")
SR1.AnlgGen.AGenChB.aPhasedSine(0).Phase("°") = 23
```

Description: Phase difference between the A and B channels of the phased sine waveform.



Waveform

On ☑ EQ □

Eq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPhasedSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChB.aPhaseSine(0).Eq
SR1.AnlgGen.AGenChA.aPhaseSine(0).Eq = False
```

Description: Sets whether the analog generator EQ file will be applied to the phased sine waveform. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPhasedSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

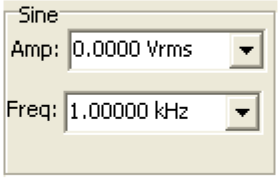
Example:

```
val = SR1.AnlgGen.AGenChB.aPhaseSine(0).On
SR1.AnlgGen.AGenChB.aPhaseSine(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.4 Sync Burst Sine

Object:	**SR1.AnlgGen.AGenCh[A/B]. aSyncBurstSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator synchronous burst sine.


Sine
Amp: 0.0000 Vrms
Freq: 1.00000 kHz

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSyncBurstSine(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aSyncBurstSine(0).Amp("Vrms") = 0
```

Description: High amplitude of the synchronous burst sine.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).Freq(*ValueUnit*)

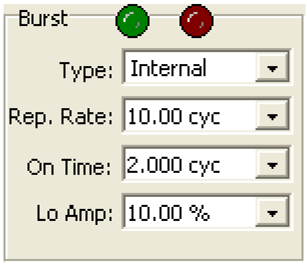
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Freq("Hz")
SR1.AnlgGen.AGenChB.aSyncBurstSine(0).Freq("Hz") = 1000
```

Description: Frequency of the synchronous burst sine.


Burst
Type: Internal
Rep. Rate: 10.00 cyc
On Time: 2.000 cyc
Lo Amp: 10.00 %

LoAmp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).LoAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).LoAmp("%")
SR1.AnlgGen.AGenChA.aSyncBurstSine(0).LoAmp("%") = 10
```

Description: Burst sine low amplitude (as a fraction of the high amplitude).

Mode

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).Mode

Command Argument(s):

Property: SineBurstType enumeration {sbInternal=0 | sbExtTriggered=1 | sbExtGated=2}

Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Mode
SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Mode = sbInternal
```

Description: Trigger mode for the synchronous burst sine.

Period

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).Period(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Period("Cyc"
SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Period("Cyc") = 10
```

Description: Burst period.

TimeOn

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).TimeOn(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSyncBurstSine(0).TimeOn("Cyc"
SR1.AnlgGen.AGenChB.aSyncBurstSine(0).TimeOn("Cyc") = 2
```

Description: On (high-amplitude) time for the burst sine.

Waveform

On ☑ EQ □

Eq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Eq
SR1.AnlgGen.AGenChA.aSyncBurstSine(0).Eq = False
```

Description: Sets whether the analog generator EQ file will be applied to the burst sine waveform. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSyncBurstSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

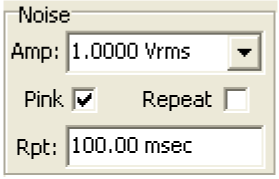
Example:

```
val = SR1.AnlgGen.AGenChA.aSyncBurstSine(0).On
SR1.AnlgGen.AGenChB.aSyncBurstSine(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.5 Noise

Object:	**SR1.AnlgGen.AGenCh[A/B].aNoise(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator noise waveform.



Noise

Amp: 1.0000 Vrms

Pink ☑ Repeat □

Rpt: 100.00 msec

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aNoise(0).Amp("Vrms")
SR1.AnlgGen.AGenChA.aNoise(0).Amp("Vrms") = 1.0
```

Description: Noise amplitude.

Pink

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).Pink

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.aNoise(0).Pink
SR1.AnlgGen.AGenChA.aNoise(0).Pink = False
```

Description: Filters the noise with a pinking filter.

RepeatPeriod

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).RepeatPeriod

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChB.aNoise(0).RepeatPeriod
SR1.AnlgGen.AGenChA.aNoise(0).RepeatPeriod = False
```

Description: Sets whether the noise waveform repeats after a specified time.

RepeatTime

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).RepeatTime(*ValueUnit*)

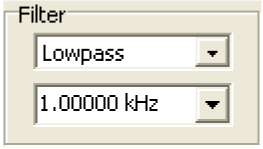
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aNoise(0).RepeatTime("s")
SR1.AnlgGen.AGenChB.aNoise(0).RepeatTime("s") = 1.0
```

Description: Sets whether the repetition period when noise repeat is on.



Filter

Lowpass

1.00000 kHz

Filter

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).Filter

Command Argument(s):

Property: NoiseFiltType enumeration {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}

Example:

```
val = SR1.AnlgGen.AGenChB.aNoise(0).Filter
SR1.AnlgGen.AGenChB.aNoise(0).Filter = nsNoFilter
```

Description: Filter type applied to the noise.

FilterFreq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).FilterFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aNoise(0).FilterFreq("HZ")
SR1.AnlgGen.AGenChB.aNoise(0).FilterFreq("HZ") = 1000
```

Description: Noise filter frequency.

Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aNoise(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

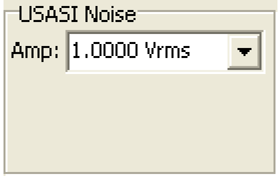
Example:

```
val = SR1.AnlgGen.AGenChA.aNoise(0).On
SR1.AnlgGen.AGenChA.aNoise(0).On = False
```

Description: Waveform on/off status.

2.4.18.1.6 USASI Noise

Object:	**SR1.AnlgGen.AGenCh[A/B].aUSASI(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator USASI noise waveform.


USASI Noise

Amp: 1.0000 Vrms

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aUSASI(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aUSASI(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aUSASI(0).Amp("Vrms") = 1.0
```

Description: Amplitude of the USASI noise waveform.

Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aUSASI(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

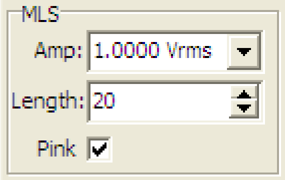
Example:

```
val = SR1.AnlgGen.AGenChB.aUSASI(0).On
SR1.AnlgGen.AGenChA.aUSASI(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.7 MLS Noise

Object:	**SR1.AnlgGen.AGenCh[A/B].aMLS(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator MLS noise waveform.



MLS
Amp: 1.0000 Vrms
Length: 20
Pink ☑

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMLS(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aMLS(0).Amp("Vrms")
SR1.AnlgGen.AGenChA.aMLS(0).Amp("Vrms") = 1.0
```

Description: Amplitude of the MLS noise waveform.

Length

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMLS(i).Length

Command Argument(s):

Property: Integer

Example:

```
val = SR1.AnlgGen.AGenChA.aMLS(0).Length
SR1.AnlgGen.AGenChB.aMLS(0).Length = 20
```

Description: Sets the length of the MLS sequence. Sequence length is 2^{length} - 1 samples.

Pink

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMLS(i).Pink

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.aMLS(0).Pink
SR1.AnlgGen.AGenChB.aMLS(0).Pink = True
```

Description: Filters the MLS noise with a pinking filter.

Waveform
On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMLS(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

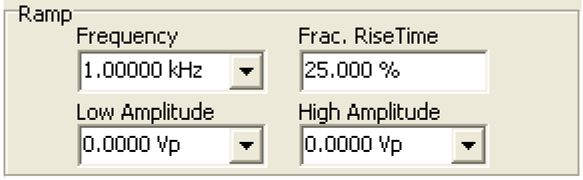
Example:

```
val = SR1.AnlgGen.AGenChB.aMLS(0).On
SR1.AnlgGen.AGenChB.aMLS(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.8 Ramp

Object:	**SR1.AnlgGen.AGenCh[A/B].aRamp(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	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: SR1.AnlgGen.AGenCh[A/B].aRamp(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aRamp(0).Freq("Hz")
SR1.AnlgGen.AGenChA.aRamp(0).Freq("Hz") = 1000
```

Description: Ramp frequency.

HiAmp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aRamp(i).HiAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aRamp(0).HiAmp("Vp")
SR1.AnlgGen.AGenChB.aRamp(0).HiAmp("Vp") = 1.0
```

Description: Maximinum ramp amplitude.

LoAmp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aRamp(i).LoAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aRamp(0).LoAmp("Vp")
SR1.AnlgGen.AGenChA.aRamp(0).LoAmp("Vp") = -1.0
```

Description: Minimum ramp amplitude.

RiseFraction

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aRamp(i).RiseFraction(*ValueUnit*)

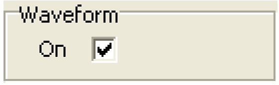
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aRamp(0).RiseFraction("%")
SR1.AnlgGen.AGenChB.aRamp(0).RiseFraction("%") = 25
```

Description: Fraction of the waveform period spent going from the low amplitude to the high amplitude.



Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aRamp(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

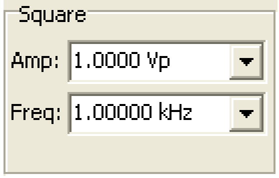
Example:

```
val = SR1.AnlgGen.AGenChA.aRamp(0).On
SR1.AnlgGen.AGenChB.aRamp(0).On = False
```

Description: On/off status of the ramp waveform.

2.4.18.1.9 Square

Object:	**SR1.AnlgGen.AGenCh[A/B].aSquare(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator square wave.



Square

Amp: 1.0000 Vp

Freq: 1.00000 kHz

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSquare(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSquare(0).Amp("Vp")
SR1.AnlgGen.AGenChB.aSquare(0).Amp("Vp") = 1.0
```

Description: Square wave amplitude.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSquare(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aSquare(0).Freq("Hz")
SR1.AnlgGen.AGenChB.aSquare(0).Freq("Hz") = 1000
```

Description: Square wave frequency.

Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aSquare(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

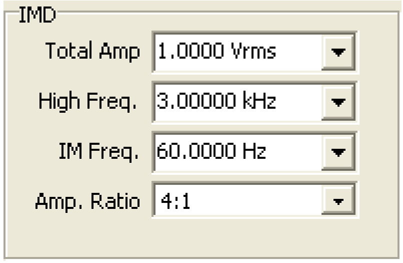
Example:

```
val = SR1.AnlgGen.AGenChA.aSquare(0).On
SR1.AnlgGen.AGenChB.aSquare(0).On = False
```

Description: On/off status of the square wave.

2.4.18.1.10 IMD

Object:	**SR1.AnlgGen.AGenCh[A/B].aIMDSig(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator IMD signal.



IMD

Total Amp: 1.0000 Vrms

High Freq.: 3.00000 kHz

IM Freq.: 60.0000 Hz

Amp. Ratio: 4:1

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aIMDSig(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aIMDSig(0).Amp("Vrms") = 1.0
```

Description: IMD waveform amplitude.

MainFreq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).MainFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aIMDSig(0).MainFreq("Hz")
SR1.AnlgGen.AGenChB.aIMDSig(0).MainFreq("Hz") = 3000
```

Description: "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: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).IMFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.AnlgGen.AGenChB.aIMDSig(0).IMFreq("Hz")
SR1.AnlgGen.AGenChA.aIMDSig(0).IMFreq("Hz") = 60
```

Description: "IM" 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

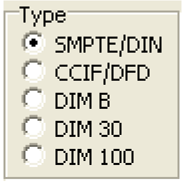
Command Syntax: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).AmpRatio

Command Argument(s):

Property: AmpRatio enumeration {ar41=0 | ar11=1}

Example:
```
val = SR1.AnlgGen.AGenChB.aIMDSig(0).AmpRatio
SR1.AnlgGen.AGenChB.aIMDSig(0).AmpRatio = ar41
```

Description: Tone amplitude ratio for the IMD signal.



Type
● SMPTE/DIN
○ CCIF/DFD
○ DIM B
○ DIM 30
○ DIM 100

Type

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).Type

Command Argument(s):

Property: IMDType enumeration {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

Example:
```
val = SR1.AnlgGen.AGenChB.aIMDSig(0).Type
SR1.AnlgGen.AGenChB.aIMDSig(0).Type = imdSMPTE
```

Description: Type of IMD waveform generated.

Waveform
On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aIMDSig(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

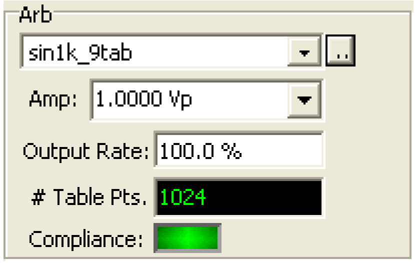
Example:

```
val = SR1.AnlgGen.AGenChA.aIMDSig(0).On
SR1.AnlgGen.AGenChA.aIMDSig(0).On = False
```

Description: On/off status of the IMD waveform.

2.4.18.1.11 Arbitrary

Object:	**SR1.AnlgGen.AGenCh[A/B].aArb(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator arbitrary waveform file.



Arb

sin1k_9tab

Amp: 1.0000 Vp

Output Rate: 100.0 %

Table Pts. 1024

Compliance: 🟩

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.AnlgGen.AGenChA.aArb(0).Amp("Vp")
SR1.AnlgGen.AGenChA.aArb(0).Amp("Vp") = 1.0
```

Description: Amplitude of the arbitrary waveform.

Compliance

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:
```
val = SR1.AnlgGen.AGenChA.aArb(0).Compliance
```

Description: Compliance status of the arbitrary waveform.

NumPointsRdg

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).NumPointsRdg

Command Argument(s):

Property: Integer (read-only)

Example:
```
val = SR1.AnlgGen.AGenChA.aArb(0).NumPointsRdg
```

Description: Number of points found in the arbitrary waveform file.

RateMultiplier

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).RateMultiplier(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aArb(0).RateMultiplier("")
SR1.AnlgGen.AGenChB.aArb(0).RateMultiplier("") = 1.0
```

Description: Output rate, in table points per output sample.

Load

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).Load(*FileName*, *ColSelect*)

Command Argument(s): *FileName* as String
ColSelect as Integer

Example:

```
Call SR1.AnlgGen.AGenChB.aArb(0).Load("v2.arb", 1)
```

Description: Loads the data in the specified column of the specified arbitrary waveform file. The maximum length is about 130,000 points if there are no other waveforms.

Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aArb(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

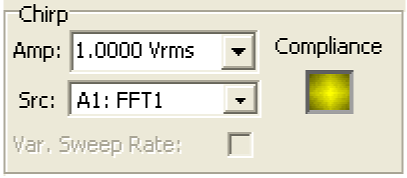
Example:

```
val = SR1.AnlgGen.AGenChA.aArb(0).On
SR1.AnlgGen.AGenChA.aArb(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.12 FFT Chirp

Object:	**SR1.AnlgGen.AGenCh[A/B].aChirp(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator fft chirp waveform.



Chirp

Amp: 1.0000 Vrms

Src: A1: FFT1

Var. Sweep Rate: □

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aChirp(0).Amp("Vrms")
SR1.AnlgGen.AGenChA.aChirp(0).Amp("Vrms") = 1.0
```

Description: Amplitude of the chirp waveform.

FFTLink

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).FFTLink

Command Argument(s):

Property: ChirpSrc enumeration {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

Example:

```
val = SR1.AnlgGen.AGenChB.aChirp(0).FFTLink
SR1.AnlgGen.AGenChA.aChirp(0).FFTLink = csA0FFT
```

Description: Linked FFT analyzer. The Chirp waveform generates a signal to match the frequency span and resolution of the linked FFT analyzer.

Compliance

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:

```
val = SR1.AnlgGen.AGenChB.aChirp(0).Compliance
```

Description: Compliance status of the chirp waveform.

FsMismatchRdg

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).FsMismatchRdg

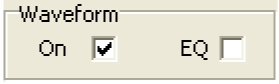
Command Argument(s):

Property: sigChirpMismatch enumeration (read-only) {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

Example:

```
val = SR1.AnlgGen.AGenChA.aChirp(0).FsMismatchRdg
```

Description: Ratio of the generator output sampling rate to the input sampling rate of the linked FFT analyzer. If the two are not integrally related scmGenMismatched is returned.



Waveform

On ☑ EQ □

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChB.aChirp(0).On
SR1.AnlgGen.AGenChB.aChirp(0).On = False
```

Description: On/off status of the waveform.

Eq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChA.aChirp(0).Eq
SR1.AnlgGen.AGenChB.aChirp(0).Eq = False
```

Description: Sets whether the current analog generator EQ file will be applied to the chirp. If EQ is on, the frequency spectrum of the chirp signal will be shaped according to the response in the EQ file.

VarSweep

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aChirp(i).VarSweep

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

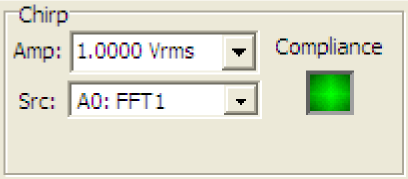
Example:

```
val = SR1.AnlgGen.AGenChA.aChirp(0).VarSweep
SR1.AnlgGen.AGenChB.aChirp(0).VarSweep = False
```

Description: Sets if variable speed sweep is enabled. If enabled (true), the crest factor of a chirp with EQ is improved.

2.4.18.1.13 Log-Sine Chirp

Object:	**SR1.AnlgGen.AGenCh[A/B].aLogSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator fft log-sine chirp waveform.



Chirp

Amp: 1.0000 Vrms

Src: A0: FFT1

Compliance 🟩

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLogSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aLogSine(0).Amp("Vrms")
SR1.AnlgGen.AGenChA.aLogSine(0).Amp("Vrms") = 1.0
```

Description: Amplitude of the log-sine chirp waveform.

FFTLink

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLogSine(i).FFTLink

Command Argument(s):

Property: ChirpSrc enumeration {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

Example:

```
val = SR1.AnlgGen.AGenChB.aLogSine(0).FFTLink
SR1.AnlgGen.AGenChA.aLogSine(0).FFTLink = csA0FFT
```

Description: Linked FFT analyzer. The Log-Sine Chirp waveform generates a signal to match the frequency span and resolution of the linked FFT analyzer.

Compliance

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLogSine(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:

```
val = SR1.AnlgGen.AGenChB.aLogSine(0).Compliance
```

Description: Compliance status of the log-sine chirp waveform.

FsMismatchRdg

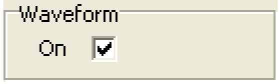
Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLogSine(i).FsMismatchRdg

Command Argument(s):

Property: sigChirpMismatch enumeration (read-only) {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

Example:
```
val = SR1.AnlgGen.AGenChA.aLogSine(0).FsMismatchRdg
```

Description: Ratio of the generator output sampling rate to the input sampling rate of the linked FFT analyzer. If the two are not integrally related scmGenMismatched is returned.



Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aLogSine(i).On

Command Argument(s):

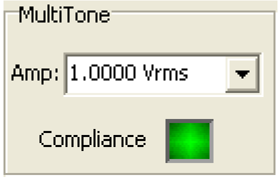
Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.AnlgGen.AGenChB.aLogSine(0).On
SR1.AnlgGen.AGenChB.aLogSine(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.14 Multitone

Object:	**SR1.AnlgGen.AGenCh[A/B].aMultiTone(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator multitone waveform. To edit the the relative amplitudes, frequencies, and phases of the tones in the multitone signal, refer to SR1.MultiToneCfg.



MultiTone

Amp: 1.0000 Vrms

Compliance

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMultiTone(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aMultiTone(0).Amp("Vrms")
SR1.AnlgGen.AGenChB.aMultiTone(0).Amp("Vrms") = 1
```

Description: Multitone amplitude.

Compliance

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMultiTone(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:

```
val = SR1.AnlgGen.AGenChA.aMultiTone(0).Compliance
```

Description: Compliance status of the multitone waveform.

ExportMultiTone

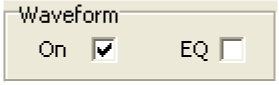
Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMultiTone(i).ExportMultiTone(*FileName*, *Mode*, *NumBits*, *Dither*)

Command Argument(s): *FileName* as String
Mode as MTExportMode enumeration {mtArbFile=0 | mtWavFile=1}
NumBits as Integer
Dither as DitherType enumeration {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

Example:

```
Call SR1.AnlgGen.AGenChA.aMultiTone(0).ExportMultiTone("
```

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: SR1.AnlgGen.AGenCh[A/B].aMultiTone(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.AnlgGen.AGenChB.aMultiTone(0).Eq
SR1.AnlgGen.AGenChB.aMultiTone(0).Eq = False
```

Description: Sets whether the current analog generator EQ file will be applied to the multitone waveform. If EQ is on, the frequency spectrum of the multitone will be modified according to the response in the EQ file.

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aMultiTone(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

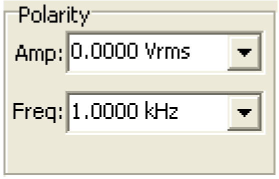
Example:

```
val = SR1.AnlgGen.AGenChB.aMultiTone(0).On
SR1.AnlgGen.AGenChA.aMultiTone(0).On = False
```

Description: On/off status of the waveform.

2.4.18.1.15 Polarity

Object:	**SR1.AnlgGen.AGenCh[A/B].aPolarity(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator polarity waveform.


Polarity
Amp: 0.0000 Vrms
Freq: 1.0000 kHz

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPolarity(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aPolarity(0).Amp("Vrms")
SR1.AnlgGen.AGenChA.aPolarity(0).Amp("Vrms") = 1
```

Description: Amplitude of the polarity waveform.

Freq

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPolarity(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChA.aPolarity(0).Freq("Hz")
SR1.AnlgGen.AGenChA.aPolarity(0).Freq("Hz") = 1000
```

Description: Frequency of the waveform.

Waveform
On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aPolarity(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

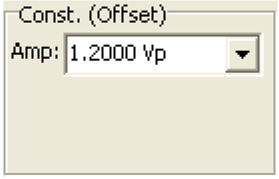
Example:

```
val = SR1.AnlgGen.AGenChA.aPolarity(0).On
SR1.AnlgGen.AGenChA.aPolarity(0).On = False
```

Description: On/off status of the polarity waveform.

2.4.18.1.16 Constant (DC)

Object:	**SR1.AnlgGen.AGenCh[A/B].aDC(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the analog generator constant waveform.



Const. (Offset)

Amp: 1.2000 Vp

Amp

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aDC(i).Amp(*ValueUnit*)

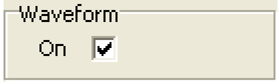
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.AnlgGen.AGenChB.aDC(0).Amp("Vp")
SR1.AnlgGen.AGenChB.aDC(0).Amp("Vp") = -1.2
```

Description: Amplitude of the DC (constant) waveform.



Waveform

On ☑

On

Command Syntax: SR1.AnlgGen.AGenCh[A/B].aDC(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

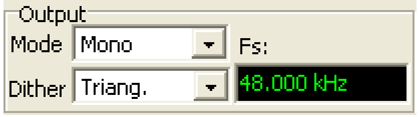
Example:

```
val = SR1.AnlgGen.AGenChA.aDC(0).On
SR1.AnlgGen.AGenChA.aDC(0).On = False
```

Description: On/off status of the waveform.

2.4.19 Digital Generator

Object:	**SR1.DigGen**
Object Argument(s):	None
Description:	Commands related to the digital generator.



Output

Mode Mono

Dither Triang.

Fs: 48.000 kHz

Mono

Command Syntax: SR1.DigGen.Mono

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.Mono
SR1.DigGen.Mono = False
```

Description: Mono/stereo mode of the digital generator.

Dither

Command Syntax: SR1.DigGen.Dither

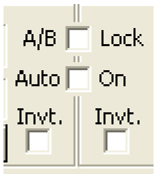
Command Argument(s):

Property: DitherType enumeration {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

Example:

```
val = SR1.DigGen.Dither
SR1.DigGen.Dither = dgNoDither
```

Description: Dither type applied to the digital audio generator.



A/B □ Lock

Auto □ On

Invt. □ Invt. □

ABLock

Command Syntax: SR1.DigGen.ABLock

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigGen.ABLock
SR1.DigGen.ABLock = False
```

Description: 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

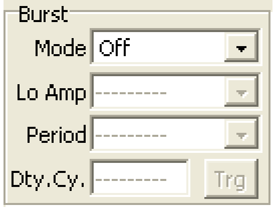
Command Syntax: SR1.DigGen.AutoOnOff

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigGen.AutoOnOff
SR1.DigGen.AutoOnOff = False
```

Description: 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.



Mode Off
Lo Amp ---------
Period ---------
Dty.Cy. ---------

BurstDutyCycle

Command Syntax: SR1.DigGen.BurstDutyCycle(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.BurstDutyCycle("pct")
SR1.DigGen.BurstDutyCycle("pct") = 25
```

Description: Fraction of the burst period that the output is in its high-amplitude state.

BurstLoAmp

Command Syntax: SR1.DigGen.BurstLoAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.BurstLoAmp("dB")
SR1.DigGen.BurstLoAmp("dB") = -20.0
```

Description: Value of the burst "low-amplitude." The value is given as a fraction of the high amplitude.

BurstMode

Command Syntax: SR1.DigGen.BurstMode

Command Argument(s):

Property: BurstType enumeration {bmNone=0 | bmTimed=1 | bmGatedHi=2 | bmGatedLo=3 | bmShaped=4 | bmTriggered=5}

Example:

```
val = SR1.DigGen.BurstMode
SR1.DigGen.BurstMode = bmNone
```

Description: Type of burst for the digital generator.

BurstPeriod

Command Syntax: SR1.DigGen.BurstPeriod(*ValueUnit*)

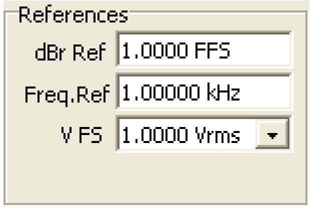
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.BurstPeriod("S")
SR1.DigGen.BurstPeriod("S") = 1.0
```

Description: Burst period.



References

dBr Ref 1.0000 FFS

Freq.Ref 1.00000 kHz

V FS 1.0000 Vrms

dBrRef

Command Syntax: SR1.DigGen.dBrRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.dBrRef("FFS")
SR1.DigGen.dBrRef("FFS") = 0.1
```

Description: Reference value for digital generator dBr units.

FreqRef

Command Syntax: SR1.DigGen.FreqRef(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.FreqRef("Hz")
SR1.DigGen.FreqRef("Hz") = 1000
```

Description: Reference frequency for digital generator relative frequency units.

VfsRef

Command Syntax: SR1.DigGen.VfsRef(*ValueUnit*)

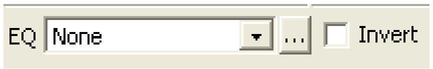
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.VfsRef("Vrms")
SR1.DigGen.VfsRef("Vrms") = 5
```

Description: "Volts Full Scale" reference.



EQ None ... □ Invert

SetEqFile

Command Syntax: SR1.DigGen.SetEqFile(*FileName*)

Command Argument(s): *FileName* as String

Example:

```
Call SR1.DigGen.SetEqFile("MyEQ.eq")
```

Description: Sets the EQ file for the digital generator.

InvertEq

Command Syntax: SR1.DigGen.InvertEq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.InvertEq
SR1.DigGen.InvertEq = False
```

Description: Invert/non-invert status for the digital generator EQ file.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.19.1 Digital Generator Channel

Object:	**SR1.DigGen.DGenCh[A/B]**
Object Argument(s):	*ch* as Integer {0=chA, 1=chB}
Description:	Commands associated with one channel of the digital audio generator.

AddWaveform

Command Syntax: SR1.DigGen.DGenCh[A/B].AddWaveform(*Type*)

Command Argument(s): *Type* as DigWaveformType enumeration {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}

Return Value: *ChanID* as Integer

Example: `ChanID = SR1.DigGen.DGenChA.AddWaveform(dwfSine)`

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: SR1.DigGen.DGenChA.Sine(ChanID)

ClearWaveforms

Command Syntax: SR1.DigGen.DGenCh[A/B].ClearWaveforms()

Command Argument(s): None

Example: `Call SR1.DigGen.DGenChB.ClearWaveforms()`

Description: Clears all waveforms from the digital generator.

DeleteWaveform

Command Syntax: SR1.DigGen.DGenCh[A/B].DeleteWaveform(*ChanID*)

Command Argument(s): *ChanID* as Integer

Example: `Call SR1.DigGen.DGenChB.DeleteWaveform(1)`

Description: Deletes the the waveform with the specified id.

SignalCount

Command Syntax: SR1.DigGen.DGenCh[A/B].SignalCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.DigGen.DGenChA.SignalCount()`

Description: Returns the current total number of waveforms configured in the digital audio generator.

SignalID

Command Syntax: SR1.DigGen.DGenCh[A/B].SignalID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *ID* as Integer

Example: `ID = SR1.DigGen.DGenChA.SignalID(2)`

Description: Returns the ID of the Indexth waveform.

SignalType

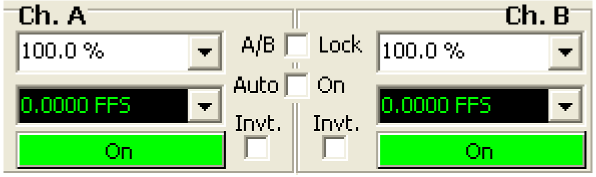
Command Syntax: SR1.DigGen.DGenCh[A/B].SignalType(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *Type* as DigWaveformType enumeration {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: `Type = SR1.DigGen.DGenChB.SignalType(3)`

Description: Returns the type of the Indexth waveform.



Gain

Command Syntax: SR1.DigGen.DGenCh[A/B].Gain(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChB.Gain("pct")
SR1.DigGen.DGenChB.Gain("pct") = 100
```

Description: Returns the gain of the specified digital audio generator channel.

Invert

Command Syntax: SR1.DigGen.DGenCh[A/B].Invert

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.Invert
SR1.DigGen.DGenChA.Invert = False
```

Description: Returns the invert/non-invert status of the digital audio generator channel.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.On
SR1.DigGen.DGenChB.On = False
```

Description: Returns the on/off status of the digital generator channel.

TotalAmpRdg

Command Syntax: SR1.DigGen.DGenCh[A/B].TotalAmpRdg(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double (read-only)

Example:

```
val = SR1.DigGen.DGenChA.TotalAmpRdg("FFS")
```

Description: Returns the total amplitude, including waveform amplitudes, EQ,channel gain, and channel on/off of the digital audio generator channel.

2.4.19.1.1 Sine

Object:	**SR1.DigGen.DGenCh[A/B].dSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator sine waveform.

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dSine(0).Amp("FFS")
SR1.DigGen.DGenChB.dSine(0).Amp("FFS") = 0.8
```

Description: Sine amplitude.

Freq

Command Syntax: SR1.DigGen.DGenCh[A/B].dSine(i).Freq(*ValueUnit*)

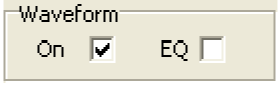
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dSine(0).Freq("Hz")
SR1.DigGen.DGenChA.dSine(0).Freq("Hz") = 10000.0
```

Description: Sine frequency.



Waveform

On ☑ EQ □

Eq

Command Syntax: SR1.DigGen.DGenCh[A/B].dSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dSine(0).Eq
SR1.DigGen.DGenChA.dSine(0).Eq = False
```

Description: Sets whether the current digital generator EQ will be applied to the sine waveform. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

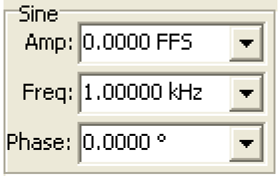
Example:

```
val = SR1.DigGen.DGenChB.dSine(0).On
SR1.DigGen.DGenChA.dSine(0).On = False
```

Description: On/off status of the sine waveform.

2.4.19.1.2 Phased Sine

Object:	**SR1.DigGen.DGenCh[A/B].dPhasedSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator phased sine waveform.



Amp: 0.0000 FFS

Freq: 1.00000 kHz

Phase: 0.0000 °

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dPhasedSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dPhasedSine(0).Amp("FFS")
SR1.DigGen.DGenChA.dPhasedSine(0).Amp("FFS") = 0.5
```

Description: Amplitude of the phased sine waveform.

Freq

Command Syntax: SR1.DigGen.DGenCh[A/B].dPhasedSine(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dPhasedSine(0).Freq("Hz")
SR1.DigGen.DGenChB.dPhasedSine(0).Freq("Hz") = 2000
```

Description: Frequency of the phased-sine waveform.

Phase

Command Syntax: SR1.DigGen.DGenCh[A/B].dPhasedSine(i).Phase(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dPhasedSine(0).Phase("")
SR1.DigGen.DGenChA.dPhasedSine(0).Phase("") = Value
```

Description: Phase difference between the A and B channels.

Eq

Command Syntax: SR1.DigGen.DGenCh[A/B].dPhasedSine(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dPhasedSine(0).Eq
SR1.DigGen.DGenChA.dPhasedSine(0).Eq = False
```

Description: Sets whether the current generator EQ file will be applied to the phased-sine waveform. If EQ is on, effective amplitude is Amp multiplied by the response of the EQ file at Freq.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dPhasedSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

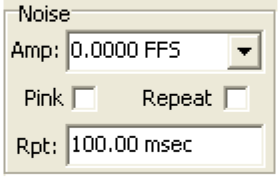
Example:

```
val = SR1.DigGen.DGenChA.dSine(0).On
SR1.DigGen.DGenChB.dSine(0).On = False
```

Description: On/off status of the phased-sine waveform.

2.4.19.1.3 Noise

Object:	**SR1.DigGen.DGenCh[A/B].dNoise(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator noise waveform.



Noise

Amp: 0.0000 FFS

Pink □ Repeat □

Rpt: 100.00 msec

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dNoise(0).Amp("FFS")
SR1.DigGen.DGenChA.dNoise(0).Amp("FFS") = 0.5
```

Description: Peak amplitude of the noise waveform.

Pink

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).Pink

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.dNoise(0).Pink
SR1.DigGen.DGenChB.dNoise(0).Pink = False
```

Description: Sets whether a pink (3 dB/oct) filter is applied to the output noise waveform.

RepeatPeriod

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).RepeatPeriod

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dNoise(0).RepeatPeriod
SR1.DigGen.DGenChB.dNoise(0).RepeatPeriod = False
```

Description: Sets whether the noise waveform repeats after a specified period.

RepeatTime

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).RepeatTime(*ValueUnit*)

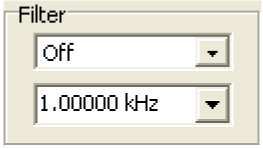
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dNoise(0).RepeatTime("s")
SR1.DigGen.DGenChB.dNoise(0).RepeatTime("s") = 0.4
```

Description: Repeat time period used when repeat is turned on.



Filter

Off

1.00000 kHz

Filter

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).Filter

Command Argument(s):

Property: NoiseFiltType enumeration {nsNoFilter=0 | nsLoPass=1 | nsHiPass=2 | nsBandPassOct3=3}

Example:

```
val = SR1.DigGen.DGenChB.dNoise(0).Filter
SR1.DigGen.DGenChB.dNoise(0).Filter = nsNoFilter
```

Description: Type of filtering applied to the noise waveform.

FilterFreq

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).FilterFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dNoise(0).FilterFreq("")
SR1.DigGen.DGenChB.dNoise(0).FilterFreq("") = Value
```

Description: Filter frequency.

Waveform

On □

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dNoise(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

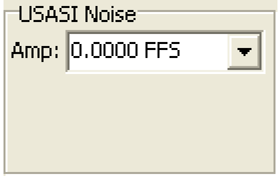
Example:

```
val = SR1.DigGen.dNoise.dSine(0).On
SR1.DigGen.DGenChA.dNoise(0).On = False
```

Description: On/off status of the noise waveform.

2.4.19.1.4 USASI Noise

Object:	**SR1.DigGen.DGenCh[A/B].dUSASI(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator USASI noise waveform.



USASI Noise

Amp: 0.0000 FFS

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dUSASI(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dUSASI(0).Amp("FFS")
SR1.DigGen.DGenChA.dUSASI(0).Amp("FFS") = 0.5
```

Description: Amplitude of the USASI noise waveform.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dUSASI(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

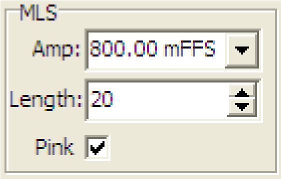
Example:

```
val = SR1.DigGen.DGenChA.dUSASI(0).On
SR1.DigGen.DGenChA.dUSASI(0).On = False
```

Description: On/off status of the USASI noise waveform.

2.4.19.1.5 MLS Noise

Object:	**SR1.DigGen.DGenCh[A/B].dMLS(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator MLS noise waveform.



MLS

Amp: 800.00 mFFS

Length: 20

Pink ☑

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dMLS(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dMLS(0).Amp("FFS")
SR1.DigGen.DGenChA.dMLS(0).Amp("FFS") = 0.8
```

Description: Amplitude of the MLS noise waveform.

Length

Command Syntax: SR1.DigGen.DGenCh[A/B].dMLS(i).Length

Command Argument(s):

Property: Integer

Example:

```
val = SR1.DigGen.DGenChA.dMLS(0).Length
SR1.DigGen.DGenChA.dMLS(0).Length = 20
```

Description: Sets the length of the MLS sequence. Sequence length is 2^{length} - 1 samples.

Pink

Command Syntax: SR1.DigGen.DGenCh[A/B].dMLS(i).Pink

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.dMLS(0).Pink
SR1.DigGen.DGenChB.dMLS(0).Pink = True
```

Description: Filters the MLS noise with a pinking filter.

Waveform

On ☑

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dMLS(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

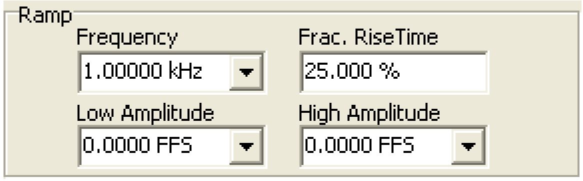
Example:

```
val = SR1.DigGen.DGenChB.dMLS(0).On
SR1.DigGen.DGenChB.dMLS(0).On = False
```

Description: On/off status of the waveform.

2.4.19.1.6 Ramp

Object:	**SR1.DigGen.DGenCh[A/B].dRamp(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator ramp waveform.



Ramp

Frequency 1.00000 kHz

Frac. RiseTime 25.000 %

Low Amplitude 0.0000 FFS

High Amplitude 0.0000 FFS

Freq

Command Syntax: SR1.DigGen.DGenCh[A/B].dRamp(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dRamp(0).Freq("Hz")
SR1.DigGen.DGenChA.dRamp(0).Freq("Hz") = 100
```

Description: Frequency of the digital ramp waveform.

HiAmp

Command Syntax: SR1.DigGen.DGenCh[A/B].dRamp(i).HiAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dRamp(0).HiAmp("FFS")
SR1.DigGen.DGenChA.dRamp(0).HiAmp("FFS") = 1.0
```

Description: High-level amplitude of the ramp.

LoAmp

Command Syntax: SR1.DigGen.DGenCh[A/B].dRamp(i).LoAmp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dRamp(0).LoAmp("FFS")
SR1.DigGen.DGenChA.dRamp(0).LoAmp("FFS") = 0.2
```

Description: Low-level amplitude of the ramp.

RiseFraction

Command Syntax: SR1.DigGen.DGenCh[A/B].dRamp(i).RiseFraction(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dRamp(0).RiseFraction("pct")
SR1.DigGen.DGenChB.dRamp(0).RiseFraction("pct") = 25
```

Description: Percentage of the total waveform period that is spent going from the low to high amplitude.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dRamp(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

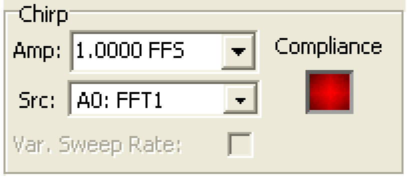
Example:

```
val = SR1.DigGen.DGenChB.dRamp(0).On
SR1.DigGen.DGenChA.dRamp(0).On = False
```

Description: On/off status of the ramp waveform.

2.4.19.1.7 FFT Chirp

Object:	**SR1.DigGen.DGenCh[A/B].dChirp(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator chirp signal.



Chirp

Amp: 1.0000 FFS

Src: A0: FFT1

Var. Sweep Rate: □

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dChirp(0).Amp("")
SR1.DigGen.DGenChB.dChirp(0).Amp("") = 0.8
```

Description: Returns the chirp amplitude.

FFTLink

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).FFTLink

Command Argument(s):

Property: ChirpSrc enumeration {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).FFTLink
SR1.DigGen.DGenChB.dChirp(0).FFTLink = csA0FFT
```

Description: Linked FFT analyzer. The Chirp waveform generates a signal to match the frequency span and resolution of the linked FFT analyzer.

Compliance

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).Compliance
```

Description: Returns the compliance status of the chirp waveform.

FsMismatchRdg

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).FsMismatchRdg

Command Argument(s):

Property: sigChirpMismatch enumeration (read-only) {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).FsMismatchRdg
```

Description: Ratio of the generator output sampling rate to the input sampling rate of the linked FFT analyzer. If the two are not integrally related scmGenMismatched is returned.

Waveform

On ☑ EQ □

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).On
SR1.DigGen.DGenChB.dChirp(0).On = False
```

Description: On/off status of the chirp waveform.

Eq

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).Eq
SR1.DigGen.DGenChB.dChirp(0).Eq = False
```

Description: Sets whether the current digital generator EQ file will be applied to the chirp waveform. If EQ is on, the frequency spectrum of the chirp will be modified according to the response in the EQ file.

VarSweep

Command Syntax: SR1.DigGen.DGenCh[A/B].dChirp(i).VarSweep

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

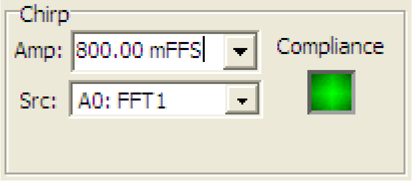
Example:

```
val = SR1.DigGen.DGenChA.dChirp(0).VarSweep
SR1.DigGen.DGenChA.dChirp(0).VarSweep = False
```

Description: Enables or disables chirp from sweeping at variable speed. If enabled (true), the crest factor of a chirp with EQ is improved.

2.4.19.1.8 Log-Sine Chirp

Object:	**SR1.DigGen.DGenCh[A/B].dLogSine(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator fft log-sine chirp waveform.


Chirp
Amp: 800.00 mFFS
Src: A0: FFT1

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dLogSine(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChA.dLogSine(0).Amp("FFS")
SR1.DigGen.DGenChA.dLogSine(0).Amp("FFS") = 0.8
```

Description: Amplitude of the log-sine chirp waveform.

dLogSine

FFTLink

Command Syntax: SR1.DigGen.DGenCh[A/B].dLogSine(i).FFTLink

Command Argument(s):

Property: ChirpSrc enumeration {csA0FFT=0 | csA0FFT2Ch=1 | csA1FFT=2 | csA1FFT2Ch=3}

Example:
```
val = SR1.DigGen.DGenChA.dLogSine(0).FFTLink
SR1.DigGen.DGenChA.dLogSine(0).FFTLink = csA0FFT
```

Description: Linked FFT analyzer. The Log-Sine Chirp waveform generates a signal to match the frequency span and resolution of the linked FFT analyzer.

Compliance

Command Syntax: SR1.DigGen.DGenCh[A/B].dLogSine(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:
```
val = SR1.DigGen.DGenChB.dLogSine(0).Compliance
```

Description: Compliance status of the log-sine chirp waveform.

FsMismatchRdg

Command Syntax: SR1.DigGen.DGenCh[A/B].dLogSine(i).FsMismatchRdg

Command Argument(s):

Property: sigChirpMismatch enumeration (read-only) {scmGenMismatched=-4 | scmGenDiv8=-3 | scmGenDiv4=-2 | scmGenDiv2=-1 | scmGenx1=0 | scmGenx2=1 | scmGenx4=2 | scmGenx8=3}

Example:

```
val = SR1.DigGen.DGenChB.dLogSine(0).FsMismatchRdg
```

Description: Ratio of the generator output sampling rate to the input sampling rate of the linked FFT analyzer. If the two are not integrally related scmGenMismatched is returned.

Waveform

On ☑

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dLogSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

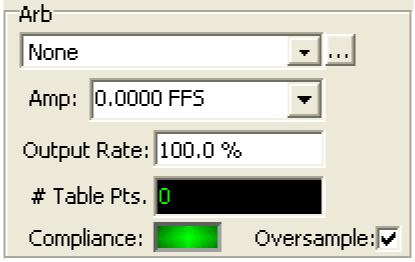
Example:

```
val = SR1.DigGen.DGenChB.dLogSine(0).On
SR1.DigGen.DGenChB.dLogSine(0).On = False
```

Description: On/off status of the waveform.

2.4.19.1.9 Arbitrary

Object:	**SR1.DigGen.DGenCh[A/B].dArb(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator arbitrary waveform.



Arb

None

Amp: 0.0000 FFS

Output Rate: 100.0 %

Table Pts. 0

Compliance: 🟩 Oversample: ☑

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChB.dArb(0).Amp("FFS")
SR1.DigGen.DGenChA.dArb(0).Amp("FFS") = 0.8
```

Description: Amplitude of the arbitrary waveform.

Compliance

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:
```
val = SR1.DigGen.DGenChA.dArb(0).Compliance
```

Description: Compliance status of the arbitrary waveform.

NumPointsRdg

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).NumPointsRdg

Command Argument(s):

Property: Integer (read-only)

Example:
```
val = SR1.DigGen.DGenChA.dArb(0).NumPointsRdg
```

Description: Number of points the arbitrary waveform file.

Oversample

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).Oversample

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.dArb(0).Oversample
SR1.DigGen.DGenChB.dArb(0).Oversample = False
```

Description: Sets whether the arbitrary waveform file points are oversampled before output. If on, output is of higher quality for non-integer output rates.

RateMultiplier

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).RateMultiplier(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dArb(0).RateMultiplier("pct")
SR1.DigGen.DGenChB.dArb(0).RateMultiplier("pct") = 100
```

Description: Output rate (in table points per sample).

Load

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).Load(*FileName*, *ColSelect*)

Command Argument(s): *FileName* as String
ColSelect as Integer

Example:

```
Call SR1.DigGen.DGenChA.dArb(0).Load("MyFile.arb", 0)
```

Description: Loads the specified column from the specified arbitrary waveform file. The maximum length is about 130,000 points if there are no other waveforms.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dArb(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

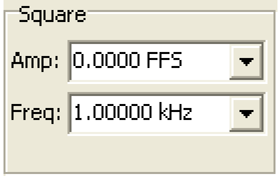
Example:

```
val = SR1.DigGen.DGenChA.dArb(0).On
SR1.DigGen.DGenChB.dArb(0).On = False
```

Description: On/off status of the arbitrary waveform.

2.4.19.1.10 Square

Object:	**SR1.DigGen.DGenCh[A/B].dSquare(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator square-wave.



Square
Amp: 0.0000 FFS
Freq: 1.00000 kHz

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dSquare(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChB.dSquare(0).Amp("")
SR1.DigGen.DGenChB.dSquare(0).Amp("") = 0.8
```

Description: Square-wave amplitude.

Freq

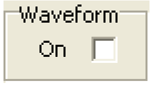
Command Syntax: SR1.DigGen.DGenCh[A/B].dSquare(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChB.dSquare(0).Freq("Hz")
SR1.DigGen.DGenChA.dSquare(0).Freq("Hz") = 1000
```

Description: Square-wave frequency.



Waveform
On ☐

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dSquare(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigGen.DGenChB.dSquare(0).On
SR1.DigGen.DGenChA.dSquare(0).On = False
```

Description: On/off status of the square-wave waveform.

PrecisionSquare

Command Syntax: SR1.DigGen.DGenCh[A/B].dSquare(i).PrecisionSquare

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

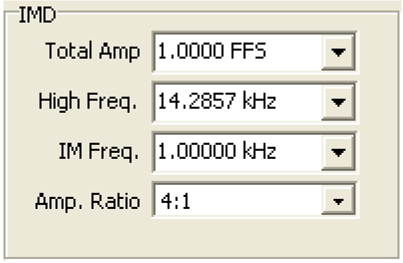
Example:

```
val = SR1.DigGen.DGenChB.dSquare(0).PrecisionSquare
SR1.DigGen.DGenChB.dSquare(0).PrecisionSquare = False
```

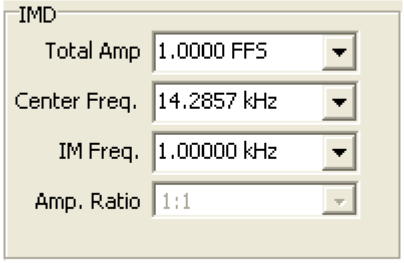
Description: Sets whether the square-wave frequency will be limited to "precision frequencies", i.e. frequencies where there are equal integer numbers of up and down samples.

2.4.19.1.11 IMD

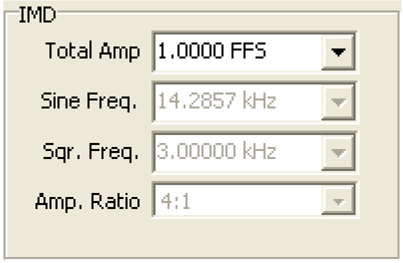
Object:	**SR1.DigGen.DGenCh[A/B].dIMDSig(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator IMD signal.



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: SR1.DigGen.DGenCh[A/B].dIMDSig(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChA.dIMDSig(0).Amp("FFS")
SR1.DigGen.DGenChB.dIMDSig(0).Amp("FFS") = 0.8
```

Description: Total amplitude of the IMD waveform.

AmpRatio

Command Syntax: SR1.DigGen.DGenCh[A/B].dIMDSig(i).AmpRatio

Command Argument(s):

Property: AmpRatio enumeration {ar41=0 | ar11=1}

Example:
```
val = SR1.DigGen.DGenChB.dIMDSig(0).AmpRatio
SR1.DigGen.DGenChB.dIMDSig(0).AmpRatio = ar41
```

Description: Tone amplitude ratio for IMD waveforms.

IMFreq

Command Syntax: SR1.DigGen.DGenCh[A/B].dIMDSig(i).IMFreq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dIMDSig(0).IMFreq("Hz")
SR1.DigGen.DGenChA.dIMDSig(0).IMFreq("Hz") = 1000
```

Description: "IM" 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: SR1.DigGen.DGenCh[A/B].dIMDSig(i).MainFreq(*ValueUnit*)

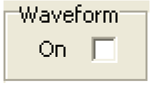
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dIMDSig(0).MainFreq("HZ")
SR1.DigGen.DGenChB.dIMDSig(0).MainFreq("HZ") = 14000
```

Description: "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.



Waveform

On □

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dIMDSig(i).On

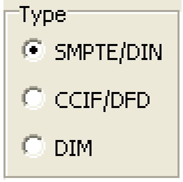
Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.dIMDSig(0).On
SR1.DigGen.DGenChB.dIMDSig(0).On = False
```

Description: On/off status of the IMD waveform.



Type

● SMPTE/DIN

○ CCIF/DFD

○ DIM

Type

Command Syntax: SR1.DigGen.DGenCh[A/B].dIMDSig(i).Type

Command Argument(s):

Property: IMDType enumeration {imdSMPTE=0 | imdCCIF=1 | imdDIMB=2 | imdDIM30=3 | imdDIM100=4}

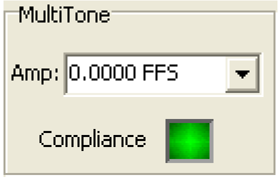
Example:

```
val = SR1.DigGen.DGenChB.dIMDSig(0).Type
SR1.DigGen.DGenChA.dIMDSig(0).Type = imdSMPTE
```

Description: Type of IMD waveform.

2.4.19.1.12 MultiTone

Object:	**SR1.DigGen.DGenCh[A/B].dMultiTone (*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator multitone signal. To edit the the relative amplitudes, frequencies, and phases of the tones in the multitone signal, refer to SR1.MultiToneCfg.



MultiTone

Amp: 0.0000 FFS

Compliance 🟩

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dMultiTone(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.DigGen.DGenChA.dMultiTone(0).Amp("FFS")
SR1.DigGen.DGenChA.dMultiTone(0).Amp("FFS") = 0.8
```

Description: .Multitone waveform amplitude.

Compliance

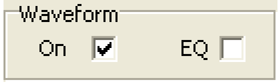
Command Syntax: SR1.DigGen.DGenCh[A/B].dMultiTone(i).Compliance

Command Argument(s):

Property: sigCompliance enumeration (read-only) {scCannotGenerate=0 | scReducedPerformance=1 | scOK=2}

Example:
```
val = SR1.DigGen.DGenChB.dMultiTone(0).Compliance
```

Description: Compliance status of the multitone waveform.



Waveform

On ☑ EQ □

Eq

Command Syntax: SR1.DigGen.DGenCh[A/B].dMultiTone(i).Eq

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigGen.DGenChB.dMultiTone(0).Eq
SR1.DigGen.DGenChA.dMultiTone(0).Eq = False
```

Description: Sets whether the currently selected EQ file will be applied to the multitone signal. If EQ is on, the frequency spectrum of the multitone will be modified according to the response in the EQ file.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dMultiTone(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChB.dMultiTone(0).On
SR1.DigGen.DGenChB.dMultiTone(0).On = False
```

Description: On/off status of the multitone waveform.

ExportMultiTone

Command Syntax: SR1.DigGen.DGenCh[A/B].dMultiTone(i).ExportMultiTone(*FileName*, *Mode*, *NumBits*, *Dither*)

Command Argument(s): *FileName* as String
Mode as MTExportMode enumeration {mtArbFile=0 | mtWavFile=1}
NumBits as Integer
Dither as DitherType enumeration {dgNoDither=0 | dgTriangular=1 | dgRectangular=2}

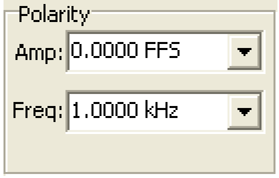
Example:

```
Call SR1.DigGen.DGenChA.dMultiTone(0).ExportMultiTone("M
```

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 SR1 arbitrary waveform file. The Dither argument specifies whether the output file will include dither.

2.4.19.1.13 Polarity

Object:	**SR1.DigGen.DGenCh[A/B].dPolarity(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator Polarity signal.



Polarity

Amp: 0.0000 FFS

Freq: 1.0000 kHz

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dPolarity(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dPolarity(0).Amp("FFS")
SR1.DigGen.DGenChB.dPolarity(0).Amp("FFS") = .8
```

Description: Amplitude of the polarity waveform.

Freq

Command Syntax: SR1.DigGen.DGenCh[A/B].dPolarity(i).Freq(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dPolarity(0).Freq("Hz")
SR1.DigGen.DGenChA.dPolarity(0).Freq("Hz") = 1000
```

Description: Frequency of the polarity waveform.

Waveform

On □

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dPolarity(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

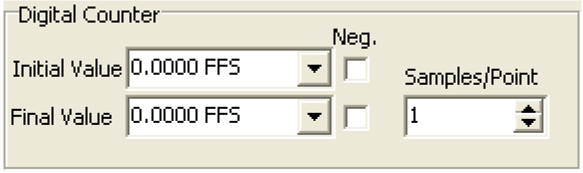
Example:

```
val = SR1.DigGen.DGenChB.dPolarity(0).On
SR1.DigGen.DGenChA.dPolarity(0).On = False
```

Description: On/off status of the polarity signal.

2.4.19.1.14 Count

Object:	**SR1.DigGen.DGenCh[A/B].dCount(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator counter waveform.



Digital Counter

Neg.

Initial Value 0.0000 FFS ☐

Final Value 0.0000 FFS ☐

Samples/Point 1

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dCount(0).Amp("FFS")
SR1.DigGen.DGenChB.dCount(0).Amp("FFS") = 0.0
```

Description: Initial value of the digital counter signal.

Polarity1

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Polarity1

Command Argument(s):

Property: SignalPolarity enumeration {Positive=0 | Negative=1}

Example:

```
val = SR1.DigGen.DGenChB.dCount(0).Polarity1
SR1.DigGen.DGenChA.dCount(0).Polarity1 = Positive
```

Description: Polarity of the initial count value.

Amp2

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Amp2(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChA.dCount(0).Amp2("FFS")
SR1.DigGen.DGenChA.dCount(0).Amp2("FFS") = 0.5
```

Description: Final value of the digital counter signal.

Polarity2

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Polarity2

Command Argument(s):

Property: SignalPolarity enumeration {Positive=0 | Negative=1}

Example:

```
val = SR1.DigGen.DGenChA.dCount(0).Polarity2
SR1.DigGen.DGenChA.dCount(0).Polarity2 = Positive
```

Description: Polarity of the final count value.

Dwell

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Dwell

Command Argument(s):

Property: Integer

Example:

```
val = SR1.DigGen.DGenChB.dCount(0).Dwell
SR1.DigGen.DGenChB.dCount(0).Dwell = 1000
```

Description: Number of samples the signal outputs each count value.

Waveform

On ☑

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

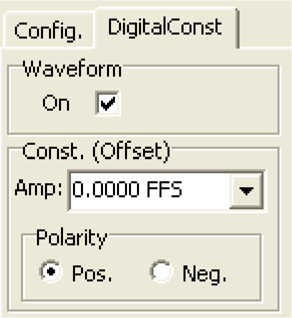
Example:

```
val = SR1.DigGen.DGenChB.dCount(0).On
SR1.DigGen.DGenChA.dCount(0).On = False
```

Description: On/off status of the digital count signal.

2.4.19.1.15 Digital Constant

Object:	**SR1.DigGen.DGenCh[A/B].dConstant(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator constant signal.



Waveform
On ☑

Const. (Offset)
Amp: 0.0000 FFS

Polarity
● Pos. ○ Neg.

Amp

Command Syntax: SR1.DigGen.DGenCh[A/B].dConstant(i).Amp(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.DigGen.DGenChB.dConstant(0).Amp("FFS")
SR1.DigGen.DGenChA.dConstant(0).Amp("FFS") = 0.334
```

Description: Amplitude of the digital constant.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dConstant(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.DigGen.DGenChA.dConstant(0).On
SR1.DigGen.DGenChB.dConstant(0).On = False
```

Description: On/off status of the digital constant signal.

Polarity

Command Syntax: SR1.DigGen.DGenCh[A/B].dConstant(i).Polarity

Command Argument(s):

Property: SignalPolarity enumeration {Positive=0 | Negative=1}

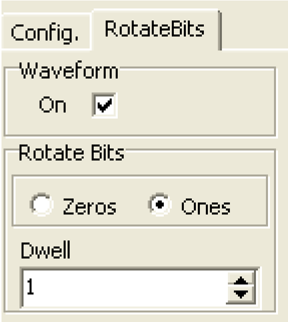
Example:

```
val = SR1.DigGen.DGenChA.dConstant(0).Polarity
SR1.DigGen.DGenChA.dConstant(0).Polarity = Positive
```

Description: Polarity of the digital constant signal.

2.4.19.1.16 Rotate

Object:	**SR1.DigGen.DGenCh[A/B].dRotate(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator rotate bits signal.



Dwell

Command Syntax: SR1.DigGen.DGenCh[A/B].dCount(i).Dwell

Command Argument(s):

Property: Integer

Example:
```
val = SR1.DigGen.DGenChA.dCount(0).Dwell
SR1.DigGen.DGenChA.dCount(0).Dwell = Value
```

Description: Number of samples the signal outputs each bit pattern for.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dSine(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:
```
val = SR1.DigGen.DGenChB.dSine(0).On
SR1.DigGen.DGenChA.dSine(0).On = False
```

Description: On/off status of the signal.

WalkVal

Command Syntax: SR1.DigGen.DGenCh[A/B].dRotate(i).WalkVal

Command Argument(s):

Property: Integer

Example:
```
val = SR1.DigGen.DGenChA.dRotate(0).WalkVal
SR1.DigGen.DGenChB.dRotate(0).WalkVal = Value
```

Description: Sets whether the signal is walking ones or walking zeros.

2.4.19.1.17 JTest

Object:	**SR1.DigGen.DGenCh[A/B].dJTest(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator J-Test signal.

J-Test

On ☑

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dJTest(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

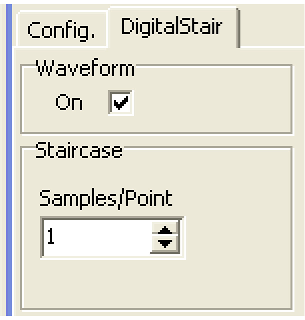
Example:

```
val = SR1.DigGen.DGenChA.dJTest(0).On
SR1.DigGen.DGenChB.dJTest(0).On = False
```

Description: On/off status of the J-Test signal.

2.4.19.1.18 Stairstep

Object:	**SR1.DigGen.DGenCh[A/B].dStair(*i*)**
Object Argument(s):	*ch* as A or B, *i* as Integer
Description:	Commands related to the digital generator stairstep signal.



Waveform

On ☑

Staircase

Samples/Point

1

Dwell

Command Syntax: SR1.DigGen.DGenCh[A/B].dStair(i).Dwell

Command Argument(s):

Property: Integer

Example:

```
val = SR1.DigGen.DGenChA.dStair(0).Dwell
SR1.DigGen.DGenChA.dStair(0).Dwell = 20
```

Description: Number of samples the stairstep signal outputs at each step value.

On

Command Syntax: SR1.DigGen.DGenCh[A/B].dStair(i).On

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

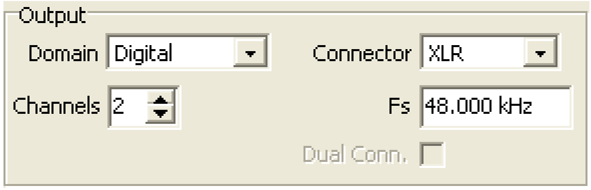
Example:

```
val = SR1.DigGen.DGenChB.dStair(0).On
SR1.DigGen.DGenChA.dStair(0).On = False
```

Description: On/off status of the stairstep signal.

2.4.20 Quick Measurmenents

Object:	**SR1.QuickMeas**
Object Argument(s):	None
Description:	Commands related to the Quick Measurements.



Output

Domain Digital

Connector XLR

Channels 2

Fs 48.000 kHz

Dual Conn. □

OutDomain

Command Syntax: SR1.QuickMeas.OutDomain

Command Argument(s):

Property: OutputDomain enumeration {odAnalog=0 | odDigital=1 | odExternal=2}

Example:

```
val = SR1.QuickMeas.OutDomain
SR1.QuickMeas.OutDomain = odAnalog
```

Description: Output domain, analog or digital, for quick measurements.

OutChannels

Command Syntax: SR1.QuickMeas.OutChannels

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.OutChannels
SR1.QuickMeas.OutChannels = 1
```

Description: Number of output channels for quick measurements.

OutConnector

Command Syntax: SR1.QuickMeas.OutConnector

Command Argument(s):

Property: IOConnector enumeration {cXLR=0 | cBNC=1 | cOptical=2}

Example:

```
val = SR1.QuickMeas.OutConnector
SR1.QuickMeas.OutConnector = cXLR
```

Description: Output connector type for quick measurements.

OutAnlgBW

Command Syntax: SR1.QuickMeas.OutAnlgBW

Command Argument(s):

Property: Bandwidth enumeration {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}

Example:

```
val = SR1.QuickMeas.OutAnlgBW
SR1.QuickMeas.OutAnlgBW = bw200kHz
```

Description: Output bandwidth selection for analog output quick measurements.

OutDigDualConn

Command Syntax: SR1.QuickMeas.OutDigDualConn

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.QuickMeas.OutDigDualConn
SR1.QuickMeas.OutDigDualConn = False
```

Description: Sets whether dual-connector mode is used when the output domain is digital.

OutDigFs

Command Syntax: SR1.QuickMeas.OutDigFs(*ValueUnit*)

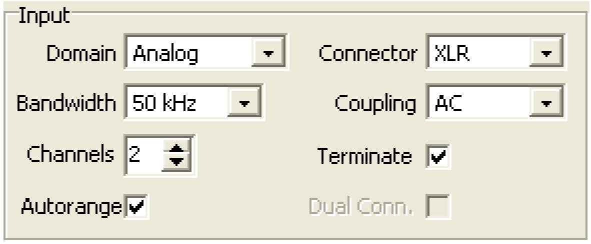
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.OutDigFs("")
SR1.QuickMeas.OutDigFs("") = 96000
```

Description: Output sample rate for digital-domain output quick measurements.



Input

Domain Analog

Connector XLR

Bandwidth 50 kHz

Coupling AC

Channels 2

Terminate ☑

Autorange ☑

Dual Conn. □

InDomain

Command Syntax: SR1.QuickMeas.InDomain

Command Argument(s):

Property: InputDomain enumeration {idAnalog=0 | idDigital=1 | idGenMon=2}

Example:

```
val = SR1.QuickMeas.InDomain
SR1.QuickMeas.InDomain = idAnalog
```

Description: Input domain (analog or digital) for quick measurements.

InChannels

Command Syntax: SR1.QuickMeas.InChannels

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InChannels
SR1.QuickMeas.InChannels = 2
```

Description: Number of input channels.

InConnector

Command Syntax: SR1.QuickMeas.InConnector

Command Argument(s):

Property: IOConnector enumeration {cXLR=0 | cBNC=1 | cOptical=2}

Example:

```
val = SR1.QuickMeas.InConnector
SR1.QuickMeas.InConnector = cXLR
```

Description: Input connector type.

InCoupling

Command Syntax: SR1.QuickMeas.InCoupling

Command Argument(s):

Property: AICPL enumeration {cplAC=0 | cplDC=1}

Example:

```
val = SR1.QuickMeas.InCoupling
SR1.QuickMeas.InCoupling = cplAC
```

Description: Input coupling mode.

InAnlgBW

Command Syntax: SR1.QuickMeas.InAnlgBW

Command Argument(s):

Property: Bandwidth enumeration {bw200kHz=0 | bw50kHz=1 | bw25kHz=2}

Example:

```
val = SR1.QuickMeas.InAnlgBW
SR1.QuickMeas.InAnlgBW = bw200kHz
```

Description: Analog bandwidth selection for analog-input quick measurements.

InAutoscale

Command Syntax: SR1.QuickMeas.InAutoscale

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.QuickMeas.InAutoscale
SR1.QuickMeas.InAutoscale = False
```

Description: Sets whether input autoranging is on or off during quick measurements.

InDigDualConn

Command Syntax: SR1.QuickMeas.InDigDualConn

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.QuickMeas.InDigDualConn
SR1.QuickMeas.InDigDualConn = False
```

Description: Sets whether dual-connector mode is used for digital input quick measurements.

InTermination

Command Syntax: SR1.QuickMeas.InTermination

Command Argument(s):

Property: Boolean enumeration {False=0 | True=1}

Example:

```
val = SR1.QuickMeas.InTermination
SR1.QuickMeas.InTermination = False
```

Description: Sets whether input termination is applied for digital input quick measurements.

Setup

Command Syntax: SR1.QuickMeas.Setup()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Setup()`

Description: Sets up the instrument according to the values entered on the Quick Measurements Setup panel.

Actions

☑ Setup
☑ SNR
☑ Reference
☑ Level

AutomMeasConfig

Command Syntax: SR1.QuickMeas.AutomMeasConfig

Command Argument(s):

Property: automMeasCfg enumeration {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:

```
val = SR1.QuickMeas.AutomMeasConfig
SR1.QuickMeas.AutomMeasConfig = 19
```

Description: Integer representing the measurements included in the automated measurement set. Each measurement is assigned a bit-value as described above. The integer is the sum of the values corresponding to the included measurements.

GetAutomMeasStatus

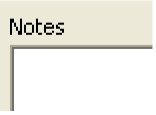
Command Syntax: SR1.QuickMeas.GetAutomMeasStatus()

Command Argument(s): None

Return Value: *Status* as AutomMeasStatus enumeration {amsIdle=0 | amsSetup=1 | amsSNR=2 | amsRef=4 | amsLevel=8 | amsTHDN=16 | amsFreqResp=32 | amsDistortion=64 | amsIMD=128 | amsCrosstalk=256 | amsInterChPhase=512}

Example: `Status = SR1.QuickMeas.GetAutomMeasStatus()`

Description: Value corresponding to the automated measurement currently being executed. If no measurement is being executed the command returns amsIdle (0).



Notes

Notes

Command Syntax: SR1.QuickMeas.Notes

Command Argument(s):

Property: String

Example:

```
val = SR1.QuickMeas.Notes
SR1.QuickMeas.Notes = "Notes for measurement."
```

Description: String corresponding to the notes added to the automated measurement report.

ClearReport

Command Syntax: SR1.QuickMeas.ClearReport()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.ClearReport()
```

Description: Clears the contents of the automated measurement report.

ExecAutomMeas

Command Syntax: SR1.QuickMeas.ExecAutomMeas()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.ExecAutomMeas()
```

Description: Begins execution of the automated measurements.

CancelAutomMeas

Command Syntax: SR1.QuickMeas.CancelAutomMeas()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.CancelAutomMeas()
```

Description: Cancels execution of the automated measurement set.

OpenReport

Command Syntax: SR1.QuickMeas.OpenReport()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.OpenReport()
```

Description: Opens the automated measurement report.

CloseReport

Command Syntax: SR1.QuickMeas.CloseReport()

Command Argument(s): None

Example: `Call SR1.QuickMeas.CloseReport()`

Description: Closes the automated measurement report.

ExportReportPDF

Command Syntax: SR1.QuickMeas.ExportReportPDF(*FileName*)

Command Argument(s): *FileName* as String

Example: `Call SR1.QuickMeas.ExportReportPDF("MyReport.pdf")`

Description: Exports the automated measurement report to the named pdf file.

UpdateReportNotes

Command Syntax: SR1.QuickMeas.UpdateReportNotes()

Command Argument(s): None

Example: `Call SR1.QuickMeas.UpdateReportNotes()`

Description: Updates the report to include the current notes string.

Automated Measurement Configuration Form Commands:

OpenAutomMeasForm

Command Syntax: SR1.QuickMeas.OpenAutomMeasForm()

Command Argument(s): None

Example: `Call SR1.QuickMeas.OpenAutomMeasForm()`

Description: Opens an automated measurement form on the current page of the page control.

OpenAutomMeasFormwID

Command Syntax: SR1.QuickMeas.OpenAutomMeasFormwID()

Command Argument(s): None

Return Value: *FormID* as Integer

Example: `FormID = SR1.QuickMeas.OpenAutomMeasFormwID()`

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: SR1.QuickMeas.CloseAutomMeasForm(*FormID*)

Command Argument(s): *FormID* as Integer

Example: `Call SR1.QuickMeas.CloseAutomMeasForm(5)`

Description: Closes the automated measurement form with the specified formID.

CloseAutomMeasForms

Command Syntax: SR1.QuickMeas.CloseAutomMeasForms()

Command Argument(s): None

Example: `Call SR1.QuickMeas.CloseAutomMeasForms()`

Description: Closes all automated measurement forms on all pages of the page control.

AutomMeasFormCount

Command Syntax: SR1.QuickMeas.AutomMeasFormCount()

Command Argument(s): None

Return Value: *Count* as Integer

Example: `Count = SR1.QuickMeas.AutomMeasFormCount()`

Description: Returns the total number of open automated measurement forms on all pages of the page control.

AutomMeasFormID

Command Syntax: SR1.QuickMeas.AutomMeasFormID(*Index*)

Command Argument(s): *Index* as Integer

Return Value: *FormID* as Integer

Example: `FormID = SR1.QuickMeas.AutomMeasFormID(Value)`

Description: Returns the formID of the Indexth automated measurement form.

Form Commands:

OpenForm

OpenFormwID

CloseForm

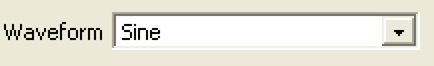
CloseForms

FormCount

FormID

2.4.20.1 Level

Object:	**SR1.QuickMeas.Level**
Object Argument(s):	None
Description:	Commands relating to the quick level measurements.



Waveform Sine

Waveform

Command Syntax: SR1.QuickMeas.Level.Waveform

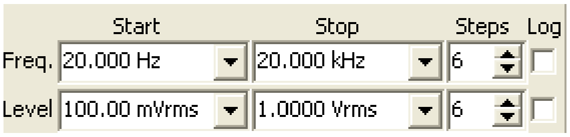
Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:

```
val = SR1.QuickMeas.Reference.Waveform
SR1.QuickMeas.Reference.Waveform = wsSine
```

Description: Generator waveform used for the level measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.Level.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Reference.FreqStart("Hz")
SR1.QuickMeas.Reference.FreqStart("Hz") = 20
```

Description: Start frequency value for the level measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.Level.FreqSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.FreqSteps
SR1.QuickMeas.SNR.FreqSteps = 6
```

Description: Number of steps in the level measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.Level.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.SNR.FreqStop("Hz")
SR1.QuickMeas.SNR.FreqStop("Hz") = 20000
```

Description: Stop frequency for the level measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.Level.FreqLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.SNR.FreqLog
SR1.QuickMeas.SNR.FreqLog = 0
```

Description: Log (1)/linear (0) status of the level measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.Level.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Reference.LevelStart("Vrms")
SR1.QuickMeas.Reference.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the level measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.Level.LevelSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.SNR.LevelSteps
SR1.QuickMeas.SNR.LevelSteps = 6
```

Description: Number of steps in the level measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.Level.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.LevelStop("Vrms")
SR1.QuickMeas.SNR.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the level measurement amplitude sweep.

LevelLog

Command Syntax: SR1.QuickMeas.Level.LevelLog

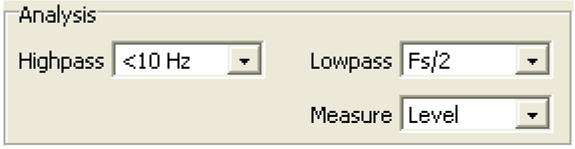
Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.LevelLog
SR1.QuickMeas.SNR.LevelLog = 0
```

Description: Log (1)/linear (0) status of the level measurement amplitude sweep.



Analysis

Highpass <10 Hz

Lowpass Fs/2

Measure Level

HighpassFilt

Command Syntax: SR1.QuickMeas.Level.HighpassFilt

Command Argument(s):

Property: HPEnum enumeration {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

Example:

```
val = SR1.QuickMeas.Reference.HighpassFilt
SR1.QuickMeas.Reference.HighpassFilt = hpDC
```

Description: Highpass filter selection for the level measurement.

LowpassFilt

Command Syntax: SR1.QuickMeas.Level.LowpassFilt

Command Argument(s):

Property: LPEnum enumeration {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

Example:

```
val = SR1.QuickMeas.Reference.LowpassFilt
SR1.QuickMeas.Reference.LowpassFilt = lpFsdiv2
```

Description: Lowpass filter selection for the level measurement.

Meas

Command Syntax: SR1.QuickMeas.Level.Meas

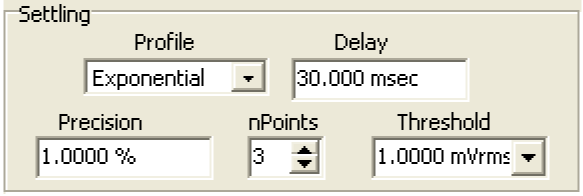
Command Argument(s):

Property: LvelRatioEnum enumeration {mLevel=0 | mRatio=1}

Example:

```
val = SR1.QuickMeas.Level.Meas
SR1.QuickMeas.Level.Meas = mLevel
```

Description: Level/ratio selection for the level measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.Level.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.SettleDelay("s")
SR1.QuickMeas.SNR.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.Level.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.SettleFloor("Vrms")
SR1.QuickMeas.SNR.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.Level.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.SNR.SettleMethod
SR1.QuickMeas.SNR.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.Level.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.SettleN
SR1.QuickMeas.SNR.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.Level.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.SettleTolerance("%")
SR1.QuickMeas.SNR.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.

FreeRun

Command Syntax: SR1.QuickMeas.Level.FreeRun()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Reference.FreeRun()`

Description: Starts the free-run level measurement.

Sweep

Command Syntax: SR1.QuickMeas.Level.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Reference.Sweep()`

Description: Starts the swept level measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.Level.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Reference.AppendTraces
SR1.QuickMeas.Reference.AppendTraces = True
```

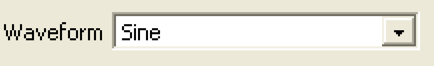
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.2 Reference

Object:	**SR1.QuickMeas.Reference**
Object Argument(s):	None
Description:	Commands related to the quick reference measurement.



Waveform Sine

Waveform

Command Syntax: SR1.QuickMeas.Reference.Waveform

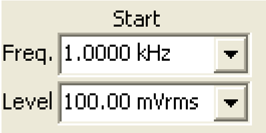
Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:

```
val = SR1.QuickMeas.Reference.Waveform
SR1.QuickMeas.Reference.Waveform = wsSine
```

Description: Generator waveform used for the reference measurement.



Start

Freq. 1.0000 kHz

Level 100.00 mVrms

FreqStart

Command Syntax: SR1.QuickMeas.Reference.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Reference.FreqStart("Hz")
SR1.QuickMeas.Reference.FreqStart("Hz") = 1000
```

Description: Frequency value for the reference measurement.

LevelStart

Command Syntax: SR1.QuickMeas.Reference.LevelStart(*ValueUnit*)

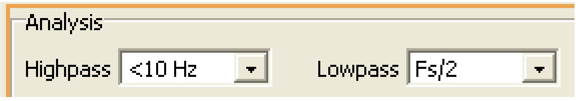
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Reference.LevelStart("Vrms")
SR1.QuickMeas.Reference.LevelStart("Vrms") = 0.1
```

Description: Frequency at which the reference measurement is made.



Highpass <10 Hz

Lowpass Fs/2

HighpassFilt

Command Syntax: SR1.QuickMeas.Reference.HighpassFilt

Command Argument(s):

Property: HPEnum enumeration {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

Example:
```
val = SR1.QuickMeas.Reference.HighpassFilt
SR1.QuickMeas.Reference.HighpassFilt = hpDC
```

Description: Highpass filter selection for the reference measurement.

LowpassFilt

Command Syntax: SR1.QuickMeas.Reference.LowpassFilt

Command Argument(s):

Property: LPEnum enumeration {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

Example:
```
val = SR1.QuickMeas.Reference.LowpassFilt
SR1.QuickMeas.Reference.LowpassFilt = lpFsdiv2
```

Description: Lowpass filter selection for the reference measurement.

FreeRun

Command Syntax: SR1.QuickMeas.Reference.FreeRun()

Command Argument(s): None

Example:
```
Call SR1.QuickMeas.Reference.FreeRun()
```

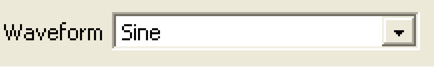
Description: Starts the free run reference measurement.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.3 SNR

Object:	**SR1.QuickMeas.SNR**
Object Argument(s):	None
Description:	Commands related to the quick Signal-to-Noise (SNR) measurements.



Waveform Sine

Waveform

Command Syntax: SR1.QuickMeas.SNR.Waveform

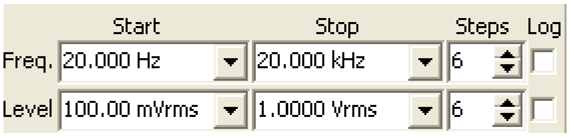
Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:

```
val = SR1.QuickMeas.SNR.Waveform
SR1.QuickMeas.SNR.Waveform = wsSine
```

Description: Generator waveform used for the SNR measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.SNR.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.FreqStart("Hz")
SR1.QuickMeas.SNR.FreqStart("Hz") = 20
```

Description: Start frequency value for the SNR measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.SNR.FreqSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.FreqSteps
SR1.QuickMeas.SNR.FreqSteps = 6
```

Description: Number of steps in the SNR measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.SNR.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.FreqStop("Hz")
SR1.QuickMeas.SNR.FreqStop("Hz") = 20000
```

Description: Stop frequency for the SNR measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.SNR.FreqLog

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.FreqLog
SR1.QuickMeas.SNR.FreqLog = 1
```

Description: Log (1)/linear (0) status of the SNR measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.SNR.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.LevelStart("Vrms")
SR1.QuickMeas.SNR.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the SNR measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.SNR.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.LevelSteps
SR1.QuickMeas.SNR.LevelSteps = 6
```

Description: Number of steps in the SNR measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.SNR.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.SNR.LevelStop("Vrms")
SR1.QuickMeas.SNR.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the SNR measurement amplitude sweep.

LevelLog

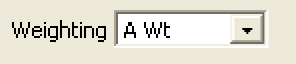
Command Syntax: SR1.QuickMeas.SNR.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.SNR.LevelLog
SR1.QuickMeas.SNR.LevelLog = 0
```

Description: Log (1)/linear (0) status of the SNR measurement amplitude sweep.



Weighting A Wt

WeightingFilt

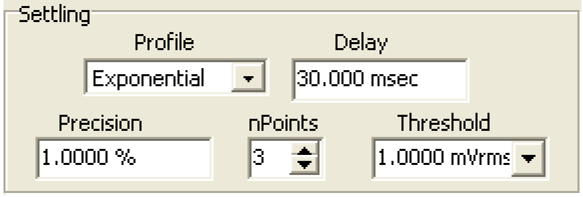
Command Syntax: SR1.QuickMeas.SNR.WeightingFilt

Command Argument(s):

Property: WeightingFilters enumeration {adNoWt=0 | adAWt=1 | adCMsg=2 | adCCITT=3 | adCCIRwtd=4 | adCCIRunwtd=5 | adCCIR2kHz=6}

Example:
```
val = SR1.QuickMeas.SNR.WeightingFilt
SR1.QuickMeas.SNR.WeightingFilt = adNoWt
```

Description: Weighting filter selection for the SNR measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.SNR.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.SettleDelay("s")
SR1.QuickMeas.SNR.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.SNR.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.SNR.SettleFloor("Vrms")
SR1.QuickMeas.SNR.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.SNR.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.SNR.SettleMethod
SR1.QuickMeas.SNR.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.SNR.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.SNR.SettleN
SR1.QuickMeas.SNR.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

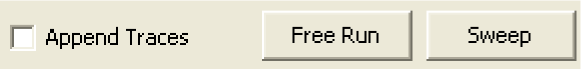
Command Syntax: SR1.QuickMeas.SNR.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.SNR.SettleTolerance("%")
SR1.QuickMeas.SNR.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.



FreeRun

Command Syntax: SR1.QuickMeas.SNR.FreeRun()

Command Argument(s): None

Example: `Call SR1.QuickMeas.SNR.FreeRun()`

Description: Starts the free-run SNR measurement.

Sweep

Command Syntax: SR1.QuickMeas.SNR.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.SNR.Sweep()`

Description: Starts the swept SNR measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.SNR.AppendTraces

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.SNR.AppendTraces
SR1.QuickMeas.SNR.AppendTraces = 0
```

Description: Sets whether new sweeps will append traces (1) to the graph or replace existing traces (0).

Form Commands:

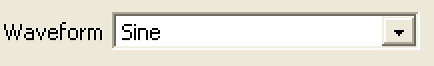
OpenForm
OpenFormwID
CloseForm
CloseForms

FormCount

FormID

2.4.20.4 THD+N

Object:	**SR1.QuickMeas.THDN**
Object Argument(s):	None
Description:	Commands related to the quick THD+N measurements.



Waveform Sine

Waveform

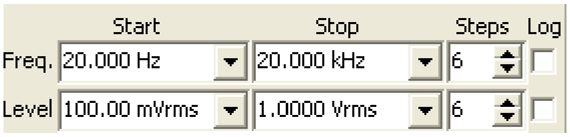
Command Syntax: SR1.QuickMeas.THDN.Waveform

Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:
```
val = SR1.QuickMeas.THDN.Waveform
SR1.QuickMeas.THDN.Waveform = wsSine
```

Description: Generator waveform used for the THD+N measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.THDN.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.THDN.FreqStart("Hz")
SR1.QuickMeas.THDN.FreqStart("Hz") = 20
```

Description: Start frequency value for the THD+N measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.THDN.FreqSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.THDN.FreqSteps
SR1.QuickMeas.THDN.FreqSteps = 6
```

Description: Number of steps in the THD+N measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.THDN.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.THDN.FreqStop("Hz")
SR1.QuickMeas.THDN.FreqStop("Hz") = 20000
```

Description: Stop frequency for the THD+N measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.THDN.FreqLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.THDN.FreqLog
SR1.QuickMeas.THDN.FreqLog = 0
```

Description: Log (1)/linear (0) status of the THD+N measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.THDN.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.THDN.LevelStart("Vrms")
SR1.QuickMeas.THDN.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the THD+N measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.THDN.LevelSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.THDN.LevelSteps
SR1.QuickMeas.THDN.LevelSteps = 6
```

Description: Number of steps in the THD+N measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.THDN.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.THDN.LevelStop("Vrms")
SR1.QuickMeas.THDN.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the THD+N measurement amplitude sweep.

LevelLog

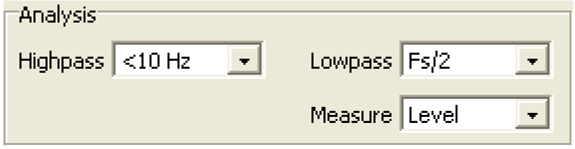
Command Syntax: SR1.QuickMeas.THDN.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.THDN.LevelLog
SR1.QuickMeas.THDN.LevelLog = 0
```

Description: Log (1)/linear (0) status of the THD+N measurement amplitude sweep.



Analysis

Highpass <10 Hz

Lowpass Fs/2

Measure Level

HighpassFilt

Command Syntax: SR1.QuickMeas.THDN.HighpassFilt

Command Argument(s):

Property: HPEnum enumeration {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

Example:
```
val = SR1.QuickMeas.THDN.HighpassFilt
SR1.QuickMeas.THDN.HighpassFilt = hpDC
```

Description: Highpass filter selection for the THD+N measurement.

LowpassFilt

Command Syntax: SR1.QuickMeas.THDN.LowpassFilt

Command Argument(s):

Property: LPEnum enumeration {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

Example:

```
val = SR1.QuickMeas.THDN.LowpassFilt
SR1.QuickMeas.THDN.LowpassFilt = lpFsdiv2
```

Description: Lowpass filter selection for the THD+N measurement.

Meas

Command Syntax: SR1.QuickMeas.THDN.Meas

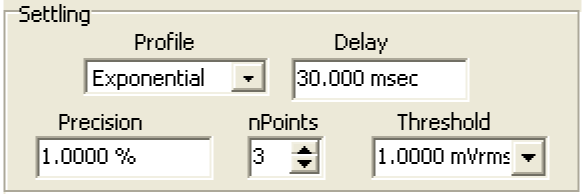
Command Argument(s):

Property: LvelRatioEnum enumeration {mLevel=0 | mRatio=1}

Example:

```
val = SR1.QuickMeas.THDN.Meas
SR1.QuickMeas.THDN.Meas = mLevel
```

Description: Level/ratio selection for the THD+N measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.THDN.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.THDN.SettleDelay("S")
SR1.QuickMeas.THDN.SettleDelay("S") = 0.030
```

Description: 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: SR1.QuickMeas.THDN.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.THDN.SettleFloor("Vrms")
SR1.QuickMeas.THDN.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.THDN.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.THDN.SettleMethod
SR1.QuickMeas.THDN.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.THDN.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.THDN.SettleN
SR1.QuickMeas.THDN.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.THDN.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.THDN.SettleTolerance("%")
SR1.QuickMeas.THDN.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.

FreeRun

Command Syntax: SR1.QuickMeas.THDN.FreeRun()

Command Argument(s): None

Example: `Call SR1.QuickMeas.THDN.FreeRun()`

Description: Starts the free-run THD+N measurement.

Sweep

Command Syntax: SR1.QuickMeas.THDN.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.THDN.Sweep()`

Description: Starts the swept THD+N measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.THDN.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.THDN.AppendTraces
SR1.QuickMeas.THDN.AppendTraces = False
```

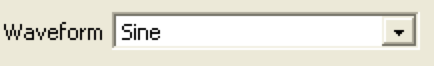
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.5 Distortion

Object:	**SR1.QuickMeas.Distortion**
Object Argument(s):	None
Description:	Commands related to the quick distortion measurements.



Waveform Sine

Waveform

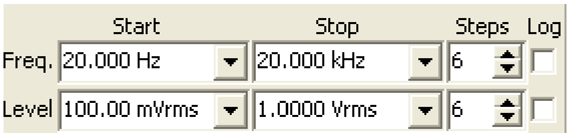
Command Syntax: SR1.QuickMeas.Distortion.Waveform

Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:
```
val = SR1.QuickMeas.Distortion.Waveform
SR1.QuickMeas.Distortion.Waveform = wsSine
```

Description: Generator waveform used for the distortion measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.Distortion.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Distortion.FreqStart("Hz")
SR1.QuickMeas.Distortion.FreqStart("Hz") = 20
```

Description: Start frequency value for the distortion measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.Distortion.FreqSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Distortion.FreqSteps
SR1.QuickMeas.Distortion.FreqSteps = 6
```

Description: Number of steps in the distortion measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.Distortion.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Distortion.FreqStop("Hz")
SR1.QuickMeas.Distortion.FreqStop("Hz") = 20000
```

Description: Stop frequency for the distortion measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.Distortion.FreqLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Distortion.FreqLog
SR1.QuickMeas.Distortion.FreqLog = 0
```

Description: Log (1)/linear (0) status of the distortion measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.Distortion.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Distortion.LevelStart("Vrms")
SR1.QuickMeas.Distortion.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the distortion measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.Distortion.LevelSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Distortion.LevelSteps
SR1.QuickMeas.Distortion.LevelSteps = 6
```

Description: number of steps in the distortion measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.Distortion.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Distortion.LevelStop("Vrms")
SR1.QuickMeas.Distortion.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the distortion measurement amplitude sweep.

LevelLog

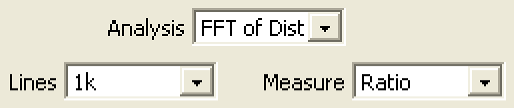
Command Syntax: SR1.QuickMeas.Distortion.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Distortion.LevelLog
SR1.QuickMeas.Distortion.LevelLog = 0
```

Description: Log (1)/linear (0) status of the distortion measurement amplitude sweep.



Analysis FFT of Dist

Lines 1k

Measure Ratio

Analysis

Command Syntax: SR1.QuickMeas.Distortion.Analysis

Command Argument(s):

Property: DistortionAnalysisEnum enumeration {daTHD=0 | daHarmonics=1 | daFFTofHarmonics=2}

Example:
```
val = SR1.QuickMeas.Distortion.Analysis
SR1.QuickMeas.Distortion.Analysis = daTHD
```

Description: Analysis method for the distortion measurement.

Meas

Command Syntax: SR1.QuickMeas.Distortion.Meas

Command Argument(s):

Property: LvelRatioEnum enumeration {mLevel=0 | mRatio=1}

Example:
```
val = SR1.QuickMeas.Distortion.Meas
SR1.QuickMeas.Distortion.Meas = mLevel
```

Description: Level/ratio setting for the distortion measurement.

FFTLines

Command Syntax: SR1.QuickMeas.Distortion.FFTLines

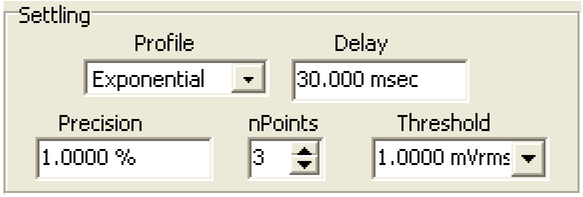
Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:

```
val = SR1.QuickMeas.Distortion.FFTLines
SR1.QuickMeas.Distortion.FFTLines = fftl32k
```

Description: FFT resolution (number of lines) for the FFT-based distortion measurement.



Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.Distortion.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Distortion.SettleDelay("s")
SR1.QuickMeas.Distortion.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.Distortion.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Distortion.SettleFloor("Vrms")
SR1.QuickMeas.Distortion.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.Distortion.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.Distortion.SettleMethod
SR1.QuickMeas.Distortion.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.Distortion.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Distortion.SettleN
SR1.QuickMeas.Distortion.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.Distortion.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Distortion.SettleTolerance("%")
SR1.QuickMeas.Distortion.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.

FreeRun

Command Syntax: SR1.QuickMeas.Distortion.FreeRun()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.Distortion.FreeRun()
```

Description: Starts the free-run distortion measurement.

Sweep

Command Syntax: SR1.QuickMeas.Distortion.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Distortion.Sweep()`

Description: Starts the swept distortion measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.Distortion.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Distortion.AppendTraces
SR1.QuickMeas.Distortion.AppendTraces = True
```

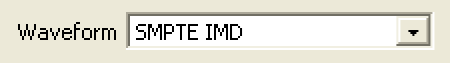
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.6 IMD

Object:	**SR1.QuickMeas.IMDistortion**
Object Argument(s):	None
Description:	Commands related to the quick IMD measurements.



Waveform: SMPTE IMD

Waveform

Command Syntax: SR1.QuickMeas.IMDistortion.Waveform

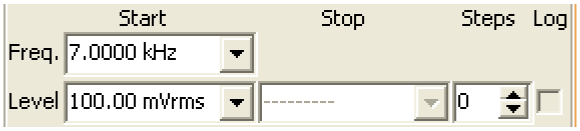
Command Argument(s):

Property: WfmIMD enumeration {wiSMPTE=0 | wiCCIF=1 | wiDIM=2}

Example:

```
val = SR1.QuickMeas.IMDistortion.Waveform
SR1.QuickMeas.IMDistortion.Waveform = wiSMPTE
```

Description: Waveform type used in the IMD measurement.



	Start	Stop	Steps	Log
Freq.	7.0000 kHz			
Level	100.00 mVrms	---------	0	□

FreqStart

Command Syntax: SR1.QuickMeas.IMDistortion.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.FreqStart("")
SR1.QuickMeas.IMDistortion.FreqStart("") = 100
```

Description: Main frequency of the IMD waveform used in the measurement.

LevelLog

Command Syntax: SR1.QuickMeas.IMDistortion.LevelLog

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.IMDistortion.LevelLog
SR1.QuickMeas.IMDistortion.LevelLog = 0
```

Description: Log (1)/linear(0) status of the IMD amplitude sweep.

LevelStart

Command Syntax: SR1.QuickMeas.IMDistortion.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.LevelStart("Vrms")
SR1.QuickMeas.IMDistortion.LevelStart("Vrms") = 0.1
```

Description: Starting amplitude for the IMD amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.IMDistortion.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.IMDistortion.LevelSteps
SR1.QuickMeas.IMDistortion.LevelSteps = 10
```

Description: Number of steps in the IMD amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.IMDistortion.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.LevelStop("Vrms")
SR1.QuickMeas.IMDistortion.LevelStop("Vrms") = 1.0
```

Description: Stop amplitude for the IMD amplitude sweep.

Products d2

Product

Command Syntax: SR1.QuickMeas.IMDistortion.Product

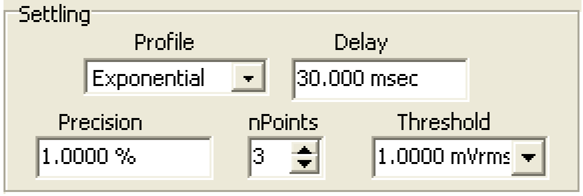
Command Argument(s):

Property: imdProducts enumeration {imdSMPTE2=0 | imdSMPTE23=1 | imdSMPTE234=2}

Example:

```
val = SR1.QuickMeas.IMDistortion.Product
SR1.QuickMeas.IMDistortion.Product = imdSMPTE2
```

Description: Distortion product selection for the IMD measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.IMDistortion.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.SettleDelay("s")
SR1.QuickMeas.IMDistortion.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.IMDistortion.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.SettleFloor("Vrms")
SR1.QuickMeas.IMDistortion.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.IMDistortion.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.IMDistortion.SettleMethod
SR1.QuickMeas.IMDistortion.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.IMDistortion.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.IMDistortion.SettleN
SR1.QuickMeas.IMDistortion.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.IMDistortion.SettleTolerance(*ValueUnit*)

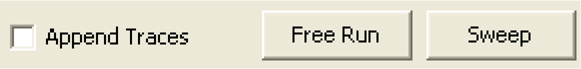
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.IMDistortion.SettleTolerance("%")
SR1.QuickMeas.IMDistortion.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.



FreeRun

Command Syntax: SR1.QuickMeas.IMDistortion.FreeRun()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.IMDistortion.FreeRun()
```

Description: Starts the free-run IMD measurement.

Sweep

Command Syntax: SR1.QuickMeas.IMDistortion.Sweep()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.IMDistortion.Sweep()
```

Description: Starts the swept IMD measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.IMDistortion.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.IMDistortion.AppendTraces
SR1.QuickMeas.IMDistortion.AppendTraces = False
```

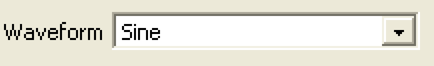
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.7 Crosstalk

Object:	**SR1.QuickMeas.Crosstalk**
Object Argument(s):	None
Description:	Commands related to the quick crosstalk measurements.



Waveform Sine

Waveform

Command Syntax: SR1.QuickMeas.Crosstalk.Waveform

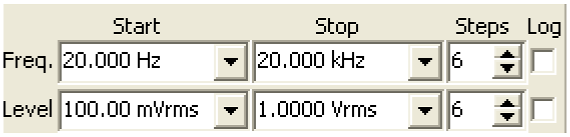
Command Argument(s):

Property: WfmSine enumeration {wsSine=0 | wsLDSine=1}

Example:

```
val = SR1.QuickMeas.Crosstalk.Waveform
SR1.QuickMeas.Crosstalk.Waveform = wsSine
```

Description: Generator waveform used for the crosstalk measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.Crosstalk.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Crosstalk.FreqStart("Hz")
SR1.QuickMeas.Crosstalk.FreqStart("Hz") = 20
```

Description: Start frequency value for the crosstalk measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.Crosstalk.FreqSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Crosstalk.FreqSteps
SR1.QuickMeas.Crosstalk.FreqSteps = 6
```

Description: Number of steps in the crosstalk measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.Crosstalk.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Crosstalk.FreqStop("Hz")
SR1.QuickMeas.Crosstalk.FreqStop("Hz") = 20000
```

Description: Stop frequency for the crosstalk measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.Crosstalk.FreqLog

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Crosstalk.FreqLog
SR1.QuickMeas.Crosstalk.FreqLog = 0
```

Description: Log (1)/linear (0) status of the crosstalk measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.Crosstalk.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Crosstalk.LevelStart("Vrms")
SR1.QuickMeas.Crosstalk.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the crosstalk measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.Crosstalk.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Crosstalk.LevelSteps
SR1.QuickMeas.Crosstalk.LevelSteps = 6
```

Description: Number of steps in the crosstalk measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.Crosstalk.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Crosstalk.LevelStop("Vrms")
SR1.QuickMeas.Crosstalk.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the crosstalk measurement amplitude sweep.

LevelLog

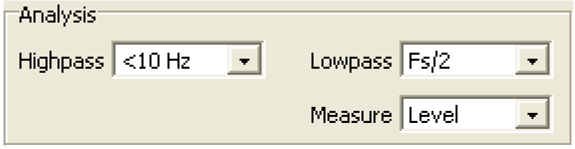
Command Syntax: SR1.QuickMeas.Crosstalk.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Crosstalk.LevelLog
SR1.QuickMeas.Crosstalk.LevelLog = 0
```

Description: Log (1)/linear (0) status of the crosstalk measurement amplitude sweep.



Analysis

Highpass <10 Hz

Lowpass Fs/2

Measure Level

HighpassFilt

Command Syntax: SR1.QuickMeas.Crosstalk.HighpassFilt

Command Argument(s):

Property: HPEnum enumeration {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

Example:
```
val = SR1.QuickMeas.Crosstalk.HighpassFilt
SR1.QuickMeas.Crosstalk.HighpassFilt = hpDC
```

Description: Highpass filter selection for the crosstalk measurement.

LowpassFilt

Command Syntax: SR1.QuickMeas.Crosstalk.LowpassFilt

Command Argument(s):

Property: LPEnum enumeration {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

Example:

```
val = SR1.QuickMeas.Crosstalk.LowpassFilt
SR1.QuickMeas.Crosstalk.LowpassFilt = lpFsdiv2
```

Description: Lowpass filter selection for the crosstalk measurement.

Meas

Command Syntax: SR1.QuickMeas.Crosstalk.Meas

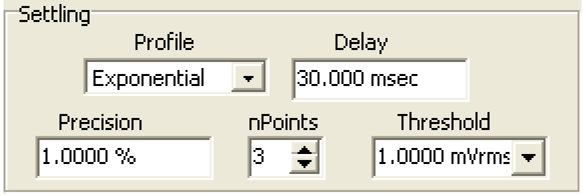
Command Argument(s):

Property: LvelRatioEnum enumeration {mLevel=0 | mRatio=1}

Example:

```
val = SR1.QuickMeas.Crosstalk.Meas
SR1.QuickMeas.Crosstalk.Meas = mLevel
```

Description: Level/ratio selection for the crosstalk measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.Crosstalk.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.Crosstalk.SettleDelay("s")
SR1.QuickMeas.Crosstalk.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.Crosstalk.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Crosstalk.SettleFloor("Vrms")
SR1.QuickMeas.Crosstalk.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.Crosstalk.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:
```
val = SR1.QuickMeas.Crosstalk.SettleMethod
SR1.QuickMeas.Crosstalk.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.Crosstalk.SettleN

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.Crosstalk.SettleN
SR1.QuickMeas.Crosstalk.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.Crosstalk.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.Crosstalk.SettleTolerance("%")
SR1.QuickMeas.Crosstalk.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.

FreeRun

Command Syntax: SR1.QuickMeas.Crosstalk.FreeRun()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Crosstalk.FreeRun()`

Description: Starts the free-run crosstalk measurement.

Sweep

Command Syntax: SR1.QuickMeas.Crosstalk.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.Crosstalk.Sweep()`

Description: Starts the swept crosstalk measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.Crosstalk.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.Crosstalk.AppendTraces
SR1.QuickMeas.Crosstalk.AppendTraces = False
```

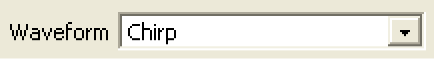
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.8 Frequency Response

Object:	**SR1.QuickMeas.FreqResp**
Object Argument(s):	None
Description:	Commands related to the quick Frequency Response measurements.



Waveform Chirp

Waveform

Command Syntax: SR1.QuickMeas.FreqResp.Waveform

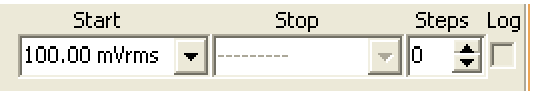
Command Argument(s):

Property: WfmChirp enumeration {wcChirp=0}

Example:

```
val = SR1.QuickMeas.FreqResp.Waveform
SR1.QuickMeas.FreqResp.Waveform = wcChirp
```

Description: Waveform used in the frequency response measurement.



Start 100.00 mVrms

Stop ---------

Steps 0

Log □

LevelStart

Command Syntax: SR1.QuickMeas.FreqResp.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.FreqResp.LevelStart("Vrms")
SR1.QuickMeas.FreqResp.LevelStart("Vrms") = 0.1
```

Description: Starting amplitude for the amplitude sweep, or the source amplitude for free-run measurements.

LevelSteps

Command Syntax: SR1.QuickMeas.FreqResp.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.FreqResp.LevelSteps
SR1.QuickMeas.FreqResp.LevelSteps = 10
```

Description: Number of steps in the amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.FreqResp.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.FreqResp.LevelStop("")
SR1.QuickMeas.FreqResp.LevelStop("") = 10
```

Description: Stopping level for the amplitude sweep.

LevelLog

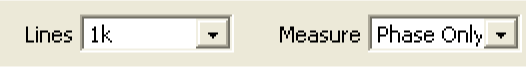
Command Syntax: SR1.QuickMeas.FreqResp.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.FreqResp.LevelLog
SR1.QuickMeas.FreqResp.LevelLog = 0
```

Description: Log(1)/linear(0) progression of the amplitude sweep.



Lines 1k

Measure Phase Only

Meas

Command Syntax: SR1.QuickMeas.FreqResp.Meas

Command Argument(s):

Property: LvelRatioEnum enumeration {mLevel=0 | mRatio=1}

Example:
```
val = SR1.QuickMeas.FreqResp.Meas
SR1.QuickMeas.FreqResp.Meas = mLevel
```

Description: Level/ratio selection of the frequency response measurement.

FFTLines

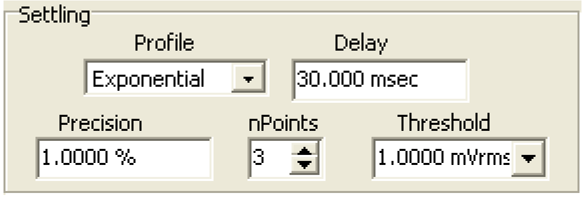
Command Syntax: SR1.QuickMeas.FreqResp.FFTLines

Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:
```
val = SR1.QuickMeas.FreqResp.FFTLines
SR1.QuickMeas.FreqResp.FFTLines = fftl32k
```

Description: Number of FFT lines (resolution) used in the frequency response measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.FreqResp.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.FreqResp.SettleDelay("s")
SR1.QuickMeas.FreqResp.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.FreqResp.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.FreqResp.SettleFloor("Vrms")
SR1.QuickMeas.FreqResp.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.FreqResp.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.FreqResp.SettleMethod
SR1.QuickMeas.FreqResp.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.FreqResp.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.FreqResp.SettleN
SR1.QuickMeas.FreqResp.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.FreqResp.SettleTolerance(*ValueUnit*)

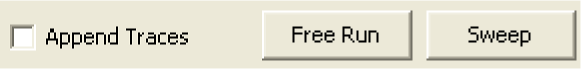
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.FreqResp.SettleTolerance("%")
SR1.QuickMeas.FreqResp.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.



FreeRun

Command Syntax: SR1.QuickMeas.FreqResp.FreeRun()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.FreqResp.FreeRun()
```

Description: Starts the free-run frequency response measurement.

Sweep

Command Syntax: SR1.QuickMeas.FreqResp.Sweep()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.FreqResp.Sweep()
```

Description: Starts the swept frequency response measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.FreqResp.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.FreqResp.AppendTraces
SR1.QuickMeas.FreqResp.AppendTraces = Value
```

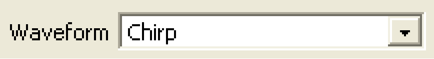
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.9 Input/Output Phase

Object:	**SR1.QuickMeas.InOutPhase**
Object Argument(s):	None
Description:	Commands related to the quick Input/Output phase measurements.



Waveform Chirp

Waveform

Command Syntax: SR1.QuickMeas.InOutPhase.Waveform

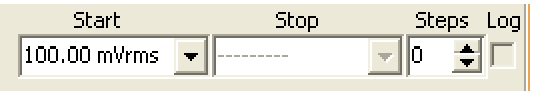
Command Argument(s):

Property: WfmChirp enumeration {wcChirp=0}

Example:

```
val = SR1.QuickMeas.InOutPhase.Waveform
SR1.QuickMeas.InOutPhase.Waveform = wcChirp
```

Description: Waveform used in the input/output phase measurement.



Start 100.00 mVrms
Stop ---------
Steps 0
Log □

LevelStart

Command Syntax: SR1.QuickMeas.InOutPhase.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InOutPhase.LevelStart("Vrms")
SR1.QuickMeas.InOutPhase.LevelStart("Vrms") = 0.1
```

Description: Starting amplitude for the amplitude sweep, or the source amplitude for free-run measurements.

LevelSteps

Command Syntax: SR1.QuickMeas.InOutPhase.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InOutPhase.LevelSteps
SR1.QuickMeas.InOutPhase.LevelSteps = 10
```

Description: Number of steps in the amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.InOutPhase.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.InOutPhase.LevelStop("")
SR1.QuickMeas.InOutPhase.LevelStop("") = 10
```

Description: Stopping level for the amplitude sweep.

LevelLog

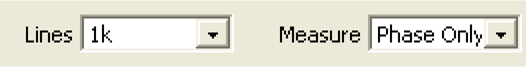
Command Syntax: SR1.QuickMeas.InOutPhase.LevelLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.InOutPhase.LevelLog
SR1.QuickMeas.InOutPhase.LevelLog = 0
```

Description: Log(1)/linear(0) progression of the amplitude sweep.



Lines 1k

Measure Phase Only

Meas

Command Syntax: SR1.QuickMeas.InOutPhase.Meas

Command Argument(s):

Property: PhaseGroupDelayEnum enumeration {mPhase=0 | mPhaseGroupDelay=1}

Example:
```
val = SR1.QuickMeas.InOutPhase.Meas
SR1.QuickMeas.InOutPhase.Meas = mPhase
```

Description: Sets whether phase or group delay will be measured.

FFTLines

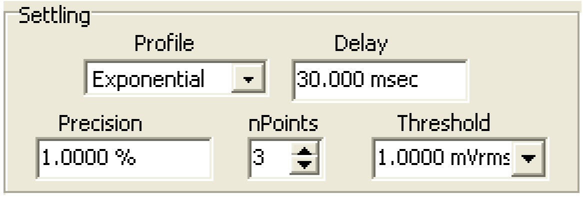
Command Syntax: SR1.QuickMeas.InOutPhase.FFTLines

Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:
```
val = SR1.QuickMeas.InOutPhase.FFTLines
SR1.QuickMeas.InOutPhase.FFTLines = fftl32k
```

Description: Number of FFT lines (resolution) used in the phase measurement.



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.InOutPhase.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InOutPhase.SettleDelay("s")
SR1.QuickMeas.InOutPhase.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.InOutPhase.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InOutPhase.SettleFloor("Vrms")
SR1.QuickMeas.InOutPhase.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.InOutPhase.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.InOutPhase.SettleMethod
SR1.QuickMeas.InOutPhase.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.InOutPhase.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InOutPhase.SettleN
SR1.QuickMeas.InOutPhase.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.InOutPhase.SettleTolerance(*ValueUnit*)

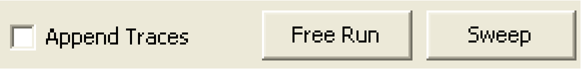
Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InOutPhase.SettleTolerance("%")
SR1.QuickMeas.InOutPhase.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.



FreeRun

Command Syntax: SR1.QuickMeas.InOutPhase.FreeRun()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.InOutPhase.FreeRun()
```

Description: Starts the free-run phase measurement.

Sweep

Command Syntax: SR1.QuickMeas.InOutPhase.Sweep()

Command Argument(s): None

Example:

```
Call SR1.QuickMeas.InOutPhase.Sweep()
```

Description: Starts the swept phase measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.InOutPhase.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InOutPhase.AppendTraces
SR1.QuickMeas.InOutPhase.AppendTraces = True
```

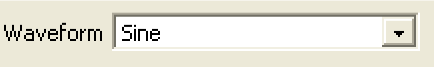
Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.20.10 InterChannel Phase

Object:	**SR1.QuickMeas.InterchPhase**
Object Argument(s):	None
Description:	Commands related to the quick interchannel phase measurement.



Waveform: Sine

Waveform

Command Syntax: SR1.QuickMeas.InterchPhase.Waveform

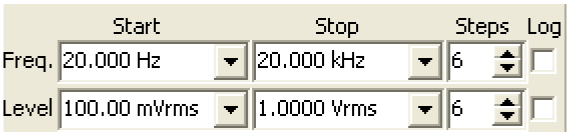
Command Argument(s):

Property: WfmSineChirp enumeration {wscSine=0 | wscLDSine=1|wscChirp=2}

Example:

```
val = SR1.QuickMeas.InterchPhase.Waveform
SR1.QuickMeas.InterchPhase.Waveform = wcChirp
```

Description: Waveform used in the interchannel phase measurement.



	Start	Stop	Steps	Log
Freq.	20.000 Hz	20.000 kHz	6	□
Level	100.00 mVrms	1.0000 Vrms	6	□

FreqStart

Command Syntax: SR1.QuickMeas.InterchPhase.FreqStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InterchPhase.FreqStart("Hz")
SR1.QuickMeas.InterchPhase.FreqStart("Hz") = 20
```

Description: Start frequency value for the phase measurement frequency sweep

FreqSteps

Command Syntax: SR1.QuickMeas.InterchPhase.FreqSteps

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.InterchPhase.FreqSteps
SR1.QuickMeas.InterchPhase.FreqSteps = 6
```

Description: Number of steps in the phase measurement frequency sweep.

FreqStop

Command Syntax: SR1.QuickMeas.InterchPhase.FreqStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.InterchPhase.FreqStop("Hz")
SR1.QuickMeas.InterchPhase.FreqStop("Hz") = 20000
```

Description: Stop frequency for the phase measurement frequency sweep.

FreqLog

Command Syntax: SR1.QuickMeas.InterchPhase.FreqLog

Command Argument(s):

Property: Integer

Example:
```
val = SR1.QuickMeas.InterchPhase.FreqLog
SR1.QuickMeas.InterchPhase.FreqLog = 0
```

Description: Log (1)/linear (0) status of the phase measurement frequency sweep.

LevelStart

Command Syntax: SR1.QuickMeas.InterchPhase.LevelStart(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:
```
val = SR1.QuickMeas.InterchPhase.LevelStart("Vrms")
SR1.QuickMeas.InterchPhase.LevelStart("Vrms") = 0.1
```

Description: Start amplitude for the phase measurement amplitude sweep.

LevelSteps

Command Syntax: SR1.QuickMeas.InterchPhase.LevelSteps

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InterchPhase.LevelSteps
SR1.QuickMeas.InterchPhase.LevelSteps = 6
```

Description: Number of steps in the phase measurement amplitude sweep.

LevelStop

Command Syntax: SR1.QuickMeas.InterchPhase.LevelStop(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InterchPhase.LevelStop("Vrms")
SR1.QuickMeas.InterchPhase.LevelStop("Vrms") = 1
```

Description: Stop amplitude for the phase measurement amplitude sweep.

LevelLog

Command Syntax: SR1.QuickMeas.InterchPhase.LevelLog

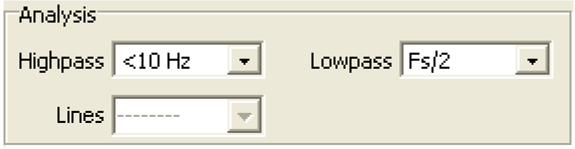
Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InterchPhase.LevelLog
SR1.QuickMeas.InterchPhase.LevelLog = 0
```

Description: Log (1)/linear (0) status of the phase measurement amplitude sweep.



Analysis

Highpass <10 Hz

Lowpass Fs/2

Lines --------

HighpassFilt

Command Syntax: SR1.QuickMeas.InterchPhase.HighpassFilt

Command Argument(s):

Property: HPEnum enumeration {hpDC=0 | hpHz22=1 | hpHz100=2 | hpHz400=3 | hpSharp400=4}

Example:

```
val = SR1.QuickMeas.InterchPhase.HighpassFilt
SR1.QuickMeas.InterchPhase.HighpassFilt = hpDC
```

Description: Highpass filter selection for the phase measurement.

LowpassFilt

Command Syntax: SR1.QuickMeas.InterchPhase.LowpassFilt

Command Argument(s):

Property: LPEnum enumeration {lpFsdiv2=0 | lpHzAES20k=1 | lpHzAES40k=2 | lpHzAES80k=3}

Example:

```
val = SR1.QuickMeas.InterchPhase.LowpassFilt
SR1.QuickMeas.InterchPhase.LowpassFilt = lpFsdiv2
```

Description: Lowpass filter selection for the phase measurement.

FFTLines

Command Syntax: SR1.QuickMeas.InterchPhase.FFTLines

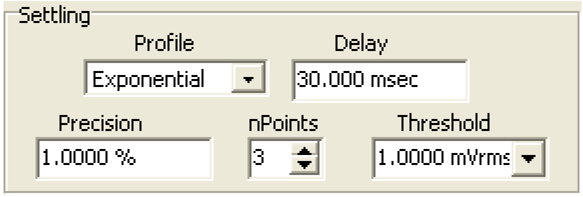
Command Argument(s):

Property: FFTlines enumeration {fftl32k=0 | fftl16k=1 | fftl8k=2 | fftl4k=3 | fftl2k=4 | fftl1k=5 | fftl512=6 | fftl256=7}

Example:

```
val = SR1.QuickMeas.InterchPhase.FFTLines
SR1.QuickMeas.InterchPhase.FFTLines = fftl32k
```

Description: Number of FFT lines (resolution) used in the interchannel phase measurement (if using chirp signal only).



Settling

Profile: Exponential

Delay: 30.000 msec

Precision: 1.0000 %

nPoints: 3

Threshold: 1.0000 mVrms

SettleDelay

Command Syntax: SR1.QuickMeas.InterchPhase.SettleDelay(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InterchPhase.SettleDelay("s")
SR1.QuickMeas.InterchPhase.SettleDelay("s") = 0.030
```

Description: 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: SR1.QuickMeas.InterchPhase.SettleFloor(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InterchPhase.SettleFloor("Vrms")
SR1.QuickMeas.InterchPhase.SettleFloor("Vrms") = 0.001
```

Description: Smallest value for the settling tolerance window.

SettleMethod

Command Syntax: SR1.QuickMeas.InterchPhase.SettleMethod

Command Argument(s):

Property: SettlingType enumeration {stlNone=0 | stlExponential=1 | stlFlat=2 | stlAverage=3 | stlSequential=4}

Example:

```
val = SR1.QuickMeas.InterchPhase.SettleMethod
SR1.QuickMeas.InterchPhase.SettleMethod = stlFlat
```

Description: Settling algorithm.

SettleN

Command Syntax: SR1.QuickMeas.InterchPhase.SettleN

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InterchPhase.SettleN
SR1.QuickMeas.InterchPhase.SettleN = 3
```

Description: Number of measurements that must meet the settling criteria before a measurement is considered settled.

SettleTolerance

Command Syntax: SR1.QuickMeas.InterchPhase.SettleTolerance(*ValueUnit*)

Command Argument(s): *ValueUnit* as String

Property: Double

Example:

```
val = SR1.QuickMeas.InterchPhase.SettleTolerance("%")
SR1.QuickMeas.InterchPhase.SettleTolerance("%") = 1.0
```

Description: Fractional size of the settling tolerance window.

FreeRun

Command Syntax: SR1.QuickMeas.InterchPhase.FreeRun()

Command Argument(s): None

Example: `Call SR1.QuickMeas.InterchPhase.FreeRun()`

Description: Starts the free-run phase measurement.

Sweep

Command Syntax: SR1.QuickMeas.InterchPhase.Sweep()

Command Argument(s): None

Example: `Call SR1.QuickMeas.InterchPhase.Sweep()`

Description: Starts the swept phase measurement.

AppendTraces

Command Syntax: SR1.QuickMeas.InterchPhase.AppendTraces

Command Argument(s):

Property: Integer

Example:

```
val = SR1.QuickMeas.InterchPhase.AppendTraces
SR1.QuickMeas.InterchPhase.AppendTraces = Value
```

Description: Sets whether new sweeps will append traces to the graph or replace existing traces.

Form Commands:

OpenForm
OpenFormwID
CloseForm
CloseForms
FormCount
FormID

2.4.21 Events

Object:	**Events**
Object Argument(s):	None
Description:	Commands related to Events. Unlike all the other SR1 objects and commands, the commands here belong to the **Events** CoClass, and are the means by which SR1 notifies the user of Scripting Events. The commands here are subroutines that the user writes to respond to various SR1 events.

General Events

General events can fire any one of five COM events, SR1Event1 thru SR1Event5. General events encompass Analog Input events (range changes, input overload), Digital Input and Status events (validity changes, emphasis changes), and Measurement events (measurement triggered, measurement has new data, digitizer analysis finished).
Multiple events can call the same COM event. They may be distinguished through the parameter passed into the COM event.
The settings of which SR1 event fires which COM event is handled in SR1.EventMgr.

SR1Event1

Event Syntax: Events.SR1Event1(*EventID*)

Event Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: None.

Example:
```
Sub Events_SR1Event1(EventID)
End Sub
```

Description: Fires when a general event has occurred.

SR1Event2

Event Syntax: Events.SR1Event2(*EventID*)

Event Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: None.

Example:

```
Sub Events_SR1Event2(EventID)
End Sub
```

Description: Fires when a general event has occurred.

SR1Event3

Event Syntax: Events.SR1Event3(*EventID*)

Event Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: None.

Example:

```
Sub Events_SR1Event3(EventID)
End Sub
```

Description: Fires when a general event has occurred.

SR1Event4

Event Syntax: Events.SR1Event4(*EventID*)

Event Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: None.

Example:

```
Sub Events_SR1Event4(EventID)
End Sub
```

Description: Fires when a general event has occured.

SR1Event5

Event Syntax: Events.SR1Event5(*EventID*)

Event Argument(s): *EventID* as EventEnum enumeration {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}

Return Value: None.

Example:

```
Sub Events_SR1Event5(EventID)
End Sub
```

Description: Fires when a general event has occured.

Limit Events

Displays have limit testing, and may trigger the following over-limit events.

BarLimitExceeded

Event Syntax: Events.BarLimitExceeded(*BarID*)

Event Argument(s): *BarID* as Integer

Return Value: None.

Example:

```
Sub Events_BarLimitExceeded(BarID)
End Sub
```

Description: Fires when the limits on a Bar display have been exceeded.
The *BarID* specifies the Bar chart via SR1.Displays.Bar(*BarID*).

GraphLimitExceeded

Event Syntax: Events.GraphLimitExceeded(*GraphID*)

Event Argument(s): *GraphID* as Integer

Return Value: None.

Example:

```
Sub Events_GraphLimitExceeded(GraphID)
End Sub
```

Description: Fires when the limits on a Graph display have been exceeded.
The *GraphID* specifies the Graph display via SR1.Displays.Graph(*GraphID*).

EyeLimitExceeded

Event Syntax: Events.EyeLimitExceeded(*DgtzrDispID*)

Event Argument(s): *DgtzrDispID* as Integer

Return Value: None.

Example:

```
Sub Events_EyeLimitExceeded(DgtzrDispID)
End Sub
```

Description: Fires when the limits on an Eye Diagram have been exceeded.
The *DgtzrDispID* specifies the Digitizer Display via SR1.Displays.DigitizerDisplay (*DgtzrDispID*).

Sweep Events

Sweeps generate several events, which the user may use to customize a sweep.

OnSweepStart

Event Syntax: Events.OnSweepStart()

Event Argument(s): None.

Return Value: None.

Example:

```
Sub Events_OnSweepStart()
End Sub
```

Description: Fires when a sweep has started.

OnSweepStep

Event Syntax: Events.OnSweepStep(*SweepCoord*)

Event Argument(s): *SweepCoord* as Array.

Return Value: None.

Example:

```
Sub Events_OnSweepStep(SweepCoord)
End Sub
```

Description: Fires when a sweep has moved to a new point.
SweepCoord is an array representing the sweep coordinate at the new sweep point. For a two-dimensional sweep, the coordinate begins at (0,0), and goes to (m,n), where n is the number of inner sweep steps, and m the number of outer sweep steps. A one-dimensional sweep will only have one sweep coordinate.

OnSweepStepTimeout

Event Syntax: Events.OnSweepStepTimeout()

Event Argument(s): None.

Return Value: None.

Example:

```
Sub Events_OnSweepStepTimeout()
End Sub
```

Description: Fires when data at a sweep point failed to settle.

OnSweepStepDone

Event Syntax: Events.OnSweepStepDone()

Event Argument(s): None.

Return Value: None.

Example:

```
Sub Events_OnSweepStepDone()
End Sub
```

Description: Fires when sweep is done collecting data at a sweep point.

OnSweepFinished

Event Syntax: Events.OnSweepFinished()

Event Argument(s): None.

Return Value: None.

Example:

```
Sub Events_OnSweepFinished()
End Sub
```

Description: Fires when a sweep has finished.

Instrument Events

OnKeypad

Event Syntax: Events.OnKeypad(*KeyCode*)

Event Argument(s): *KeyCode* as Integer.

Return Value: None.

Example:

```
Sub Events_OnKeypad(KeyCode)
End Sub
```

Description: Fires when an SR1 front-panel key is pressed.
Each key has its own numeric code.

OnKnob

Event Syntax: Events.OnKnob(*KnobCode*)

Event Argument(s): *KnobCode* as Integer.

Return Value: None.

Example:

```
Sub Events_OnKnob(KnobCode)
End Sub
```

Description: Fires when the SR1 front-panel knob has been turned.
Larger codes represent greater movement, and negative codes represent counter-clockwise movement.

OnWarning

Event Syntax: Events.OnWarning(*Error*)

Event Argument(s): *Error* as String.

Return Value: None.

Example:

```
Sub Events_OnWarning(Error)
End Sub
```

Description: Fires when an SR1 error occurs.

OnCriticalError

Event Syntax: Events.OnCriticalError(*Error*)

Event Argument(s): *Error* as String.

Return Value: None.

Example:

```
Sub Events_OnCriticalError(Error)
End Sub
```

Description: Fires when a critical SR1 error occurs. The instrument will quit on a critical error.

OnScriptError

Event Syntax: Events.OnScriptError(*Error*)

Event Argument(s): *Error* as String.

Return Value: None.

Example:

```
Sub Events_OnScriptError(Error)
End Sub
```

Description: Fires when a local Scripting error occurs.

UserEvent

Event Syntax: Events.UserEvent()

Event Argument(s): None.

Return Value: None.

Example:

```
Sub Events_UserEvent()
End Sub
```

Description: Fires when it is manually triggered in a user script by calling SR1.EventMgr. FireUserEvent().

SRS Stanford Research Systems